TRICKY TUTORIAL # 5

Have you ever wondered how THEY program all those great games you see in the arcades? Now you will be able to study all the basics of constructing animated games on the ATARI(tm) computers. This Tutorial includes 14 main programs with over 25 examples, including a Color Selection Utility, A Playfield editor, and a Player Missile Shape editor. Some of the topics explained in the 50 page manual and in the included programs are:

Player Shapes
Priority
Collisions
Reserving Memory
Single/Double Line Resolution
How to Animate
Size Control
A COMPLETE GAME EXAMPLE
TWO MACHINE LANGUAGE UTILITIES

The program will run in 16k tape machines except for the Playfield and Player utilities, which are not necessary to enjoy and use the Tutorial. These utilities need 32k of memory. As always, you have our money back guarantee!

SANTA CRUZ

EDUCATIONAL SOFTWARE

Presents

TRICKY TUTORIAL # 5

PLAYER MISSILE GRAPHICS

by

ROBIN ALAN SHERER

AND

BILL BRYNER

- TRICKY TUTORIAL #5 -

****

PLAYER MISSILE GRAPHICS

Robin Alan Sherer
and
Bill Bryner

****

Preliminary Notes For Tape Users

Before I begin, some special notes are needed to help tape users and those with only 16K of memory. I got a bit carried away with the number of program examples that are included in such a low cost product. I either had to go to two tapes to hold them all, and consequently raise the price (Heaven forbid!!), or leave a few out. I chose the latter. Consequently some of you may have trouble reading in the programs on the 410 tape player. Before you start, wind the tape player forward and backward once. To load, reset your recorder's counter to 0. Now type RUN "C:" and press return twice. The tape should begin to load into memory. If loading any of the examples produces an error message, you have two options:

1) Rewind the tape to 0, then take it out and listen to it on a regular cassette Player. The point you want to find is the beginning of the steady tone that precedes the digital signal. This tone will continue for about 20 seconds before the actual program begins to load. If you are not sure if the tape is at the beginning, just rewind or advance it a little bit until you find a place where there is no steady tone, then move back to the start of the tone. Now place it back into the 410 recorder and reset the counter to 0. The tape should load now. As soon as you get a successful load, save each program on another backup cassette. This is a good habit to develop when using cassettes.

For most of our tape software, you only have to find the starting place once. This tape contains 14 programs, making it very difficult to maintain the proper spacing between programs while mass producing them at a low cost. Besides, programs saved by your own machine will play back more reliably at your home than when recorded by a high-speed duplicator.

2) You can "guess" the correct spot, going back and forth until you get lucky (very frustrating).

If any program doesn't load properly, you should try another recorder (perhaps at a store or a friend's house). If you are so upset at the darn thing you want to buy a disk drive, send it back for a prompt replacement to:

S.C. Educational Software
5425 Jigger Dr.
Soquel, Ca. 95073
(408) 476-4901

****

- 16K USERS -

I haven't forgotten you. All programs on this tape except the last two utilities will run on your machines. The utilities are very nice, but not necessary, so just save them until you increase your memory size. The last program you will be able to read is the one that has the ATARI logo as an example of two players side by side. (Ex. 5.13)

To advance to the next program: Press the OPTION key in order to advance to the next example. During the Lunar Lander program, choose a point when the sound effects are least audible and hold the OPTION key down ten to fifteen seconds or however long it takes for the next program to begin to load.

- HOW TO USE THIS MANUAL -

Without the information in this manual, these programs are rather boring. Only when you understand what is being shown in the examples will Player Missile Graphics (PMG) begin to make sense to you. In fact, after you have written your first real game or business application you may just feel a wonderful sense of accomplishment!

As you progress into the manual, you may find yourself lost on a certain subject. IMMEDIATELY refer back to the following chart to see which examples demonstrate the feature in question. Note that I refer to Example 5.1 as 1 since we all know this is Tutorial #5. When the chart says "others" this means that the other examples demonstrate the topic also, but not as well as those listed. Here then is ...

- THE FOLLOWING CHART -

TOPIC	SEE EXAMPLES
**********	**********
General Features of PMG	1,2,14
Playfields (backgrounds)	2,4,8
Color Selection & Pokes	5

TOPIC	SEE EXAMPLES
Memory Area for PMG	7 + others
How to Draw a Player	6,7,14, others
Missiles Used as a Player	5,14
Joystick or Paddle?	7
Use of Sound with PMG	1,2,8
Moving Players and Missiles	1,2,6,7,9
Size of Players and Missiles	6,14
Single/Double Line Resolution	10
How to Animate A Player	2,11,12
Priority	2,12
Collisions	2,12
Two Players as One	13
Complete Game	2
PMG Using Strings	4,14

****

GENERAL DISCUSSION

OK! You spent your hard-earned money on this program because everyone says that Player Missile Graphics is the way to go, but you know it is too hard to ever really understand. Besides, you bought $25 worth of magazines for the two-page articles on the subject, and they all said the same thing (which made little sense to you). After all that, you got ATARI's book, DE RE ATARI, which was great reading, but Chris Crawford only explains the basic capabilities of PMG. How do you put it all to practical use? Well...

WE'RE GOING TO WRITE A GAME!

Most of us like to play arcade games, and since most arcade games use cute little figures and a maze, we will design our game that way. Currently the favorite game of many people is called PACMAN (tm). We can't actually use the same game as the arcades due to legal problems, but we can take the cute little character and put him into our own hair-raising situation. All games take imagination. Let's put ours to work making up all kinds of shapes to

animate and "play" with on the screen. Some of these will be used in demos within the 14 main programs. You can take over where these lessons leave off and create any kind of application you wish. Player missiles are not just for games. Any time you need color or movement in your programs, PMG is the way to go. For example, ATARI's scram program, a Nuclear Plant Simulator, uses players and missiles for the various tanks and mechanisms on the screen.

WHAT IS A PLAYER/MISSILE?

The names "Players and Missiles" came from the original use of these objects as guns (for the Player to use) and bullets or missiles for the "shot" that was fired. Now, of course, many other uses exist, but the names are still used. Let's begin by looking at the original use of PMG, a space game demonstration. Run the first program now and follow along in the discussion...

****

-PROGRAM 5.1-

If you are new to the ATARI computer, I suggest you watch this example many times and refer to it later as each subject is mentioned. Here is what you are seeing. First, the screen on your TV will go black. This is just due to turning off the display processor, called ANTIC, which accomplishes two things:

1) It speeds up the machine by 30%
2) It looks nicer to have the background of stars suddenly appear, rather than see it slowly drawn out

You can compare this method to actually watching the background drawn out in the next program and choose which method you prefer. If you decide to turn off the screen display while YOUR backgrounds are drawn, here's how!

At the point you want the screen to "turn off":

```
100   ON = PEEK (559): POKE 559,0
 .
 .        (PROGRAM CODE TO DRAW YOUR BACKGROUND)
 .
500   POKE 559, ON
```

Later, I'll teach you what the correct number for "ON" is, but you can always just save the old value as I did above. The number in memory location 559 will have to be changes later to allow PMG to work.

TWO PROCESSORS?

Yes, the ATARI does have two microprocessors. The ANTIC is a special one that is used to draw the screen on your TV or Monitor. It does both graphics and text. Whenever it is in use, it "steals cycles" from the main microprocessor, the 6502. This is why the computer is faster when the screen is blank (559=0)...the main processor runs at full speed if it doesn't have to stop all the time to allow the ANTIC time to draw the screen.

Are you still in Program 5.1? Good, don't rush ahead. This is going to take time to explain properly. OK, now the computer has a nice bunch of "stars" on the screen (use your imagination). Some are red and some blue. This is because when we plot single points in Graphics 8, the TV can only light up either a blue or a red pixel on the screen. If you look really closely at your tube you will see these tiny pixels of red and blue. These background stars are called a Playfield when discussing PMG. If you look at the graphics chart on page 53 of your basic manual, you'll see that most graphics modes have from one to four colors allowed. Modes 9 to 11 have more, but still only four of the colors pertain to the following discussion.

When you say color 1, and then draw out some shapes on the screen, you are drawing what we call PLAYFIELD 1. Likewise, if you use color 2 it is Playfield 2, if you use color 3 it is Playfield 3. The fourth Playfield is actually color 0, the background color that you get when you start out in a graphics mode. All of these colors are under your control, and I will explain how to use them presently.

The border is obvious, but notice the control over its color. I usually set its color to the background Playfield color so the whole screen appears the same color. Notice how the stars don't actually go to the edge of the screen, but since the players can move anywhere, having the border the same color makes the Playfield seem larger.

At last we have two players on the screen. It has been a long-winded discussion, but there they are. And, of my! They move! Oh, you knew they could. Well, I guess there is no surprising you. Notice how easily basic can move them HORIZONTALLY across the screen. Later, when you look at the listing for this program, notice 5 lines (400 to 440) control all the sound and movement (You are going to study the code, aren't you?). Next, the players grow in size. This capabilities also is a simple poke of a number, as will be explained later.

Finally, our space battle concludes with one ship firing missiles at another until they both are destroyed and the example is over. Run the example again until you are clear as to what the following are:

A PLAYER
A MISSILE
THE BACKGROUND
PLAYFIELDS 0 TO 3
A BORDER
SIZE CHANGES

****

OUR COMPLETE GAME ALREADY?

Rather than talk about some strange ideas like collision registers now, I thought you would like to see and play with the goal of this Tricky Tutorial, an actual game using all of the features of PMG. Now before you begin, please realize that our goal did not include producing a game just like the arcades. Those are written in machine language with special display tubes to allow very fine graphics. We just want to demonstrate all of the topics you will be

learning later in one program. For the hot programmers in the audience, the game can be expanded to be quite a lot of fun. As it is now, it just fits in 16k which was required for our friends with ATARI 400's (and there are a lot of you out there!).

The "rules" I am about to present are made up, and thus can be changed easily. In fact, the first thing you might do when you finish the complete lesson is to change the game to your own specifications. Most changes will be surprisingly simple. Until then, here are my rules:

HOW TO PLAY THE GAME

Plug in joysticks into ports 1 and 2. Each Player moves his character and tries to score points. One point is scored each time you "gobble" one of the two "Energizer Pellets" that will appear from time to time. Five points are scored if you gobble up your opponent. This is done by first touching the blue recharge zone in the center of the maze. Your character will change color and may score by touching the other Player. If the other Player touches the recharge zone, then he can get you and your color changes back to normal.

I started the scores out at three each, just to point out that everything doesn't have to begin at 0. The logic to the game is to balance your moves between going for easy single pointers and the big score of five points!

THE GAME AIN'T SO HOT!

That's true, but think of this exciting fact. All of the basic techniques of the way most real arcade games are written are included in this little game. The only improvements needed are your own ideas and SPEED. arcade games are written in machine language which accounts for the speed. The ideas are up to you!

While using the game notice the way the characters move. They are slow, yet I am moving them with machine language routines (these will be explained later). What slows them down is the Basic language needed to call them. Also notice what happens when the characters touch the walls of the maze. After you feel you have a real idea of what features are in the game, run example 5.3 and we'll begin the lesson.

****

PLAYFIELDS - EX. 5.3

This example is so simple to understand that my daughter even helped me write it. The basic idea I already mentioned above, but I'll repeat myself just this once (or twice). The stuff you see on the screen that is not a Player or Missile is called PLAYFIELD. Playfields are created by plotting points using the PLOT and DRAW to commands, or by direct POKE-ing of data into the memory area that is displayed on the screen. You should already be familiar with PLOT and DRAW to from your Basic manual. If not, read about them in the manual and then come back.

PRESS 1 on your keyboard and plug in a joystick into port #1. Move the joystick around to draw color 1 on the screen. Now PRESS 2 and draw with

color 2. PRESS 3 and do the same. Finally PRESS 0 and draw with color 0. Notice that color 0 seems to erase rather than draw, but I assure you it is drawing. Think about it. Color 0 draws with the background color so when it is plotted any other color of course is changed to the background color (I know this was obvious to some of you).

The purpose of the above exercise was to demonstrate the playfields. Playfield 0 is the background color; Playfield 1 is whatever is drawn with color 1, etc. Page 53 of your Basic manual shows which Graphics modes allow which playfields. Although very simple, this example is important for you since you must know which Playfield you are testing for when you use the collision registers discussed later.

****

- A NOTE ON COLORS -

Throughout all of these examples, if you don't like the colors being used, please change them to match your TV. I have several sets that all show the colors differently, so some of these examples may look strange to you. Here is how to change the colors:

PLAYFIELD COLORS

COLOR #	POKE LOCATION
0	710
1	708
2	709
3	711
Border & text window	712

The correct value to poke is:

```
COLOR * 256 + LUMINANCE
```

where color is from 0 to 15 and luminance is from 0 to 14 (even numbers).

Look within any example you want to change until you find the correct number being poked and place your color choice in the Poke statement.

****

SURPRISE #1

EX. 5.4 (Playfield Editor)

You probably realize that any program, whether for a game or a business application, will need a background. The exception is a text only program which would seldom need the use of PMG. This means we need a nice way to create shapes and plot them on the screen. Let's say you want a space game with mountains, enemy bases, a few ships on the ground to shoot at, and maybe even some trees. These need to be plotted on the screen before the game can

begin. The normal method is to use graph paper and hand calculate each point to be plotted. WOW! That is the hard way. Feeling inspired, Bill and I decided to whip up a little editor to help you. Here is how to use it.

NOTE

Because this program requires 32k to run, it is the next to the last program on tapes. For disk users it is in the same order we have been discussing.

When the program comes up, disk users may press "L" and then "D" to bring in a previously saved drawing as a test. You will notice a flashing dot, or cursor, on the screen. Tape users will have to create and save a drawing before the LOAD option will work. This is the point that is being drawn as you move it. You can choose between playfields 0 to 3. First decide how many playfields the Graphics mode you are going to use allows. For example, if your program will use Graphics 5 you have all four colors to work with, but if Graphics 6, then you only have two. Press the number of the PF color and use the cursor keys (don't press CTRL) to move the cursor. Remember to erase, you plot in the background, color 0. Draw out a building or spaceship. When done, the editor allows you to save the shape two ways:

1) To save the shapes for future use or modifications, you may press "S" and then "T" or "D" for tape or disk. Tape users should be sure to record the location on the tape where the program starts so that they may find the data again. A blank tape is recommended. Disk users will always find the data saved under the same name on the disk, PFDATA, so if you want to keep the shapes, use the rename command "E", FROM DOS, to name the data something more meaningful.

2) To use the shapes in your program, you will need a simple subroutine. You may look in the program code for this example to see a very sophisticated way to handle the data using strings. Feel free to copy parts of it for your own use. Here, however, is the simple way to plot the data.

Before presenting the actual code, let me remind you what we are doing. Each color of your shape needs to be plotted separately. Also, you will probably want to plot the shape (for ex: a tree with some ground around it) at several points in the background. Finally, you will have several other shapes like men, spaceships, and planets that must also be plotted.

To get the data for your shape, just press SELECT and wait for 10 seconds. The text area of the screen will start presenting DATA statements that are the points to plot. The upper left corner of the square that limits your shape, if colored, would be 1,1. These numbers represent first an X value, and than a Y value of a standard grid system such as the ATARI normally uses for Graphics. To see all of the data for each color, press START each time the data stops. If you have more than 199 points in your shape of any one color, the program will say "TOO MUCH DATA". Bill and I could only fit in that many points into a 32K program, so go back and change a few points to a different color. It won't happen for most shapes. The numbers of the DATA statements should be changed to fit within your program. You may want trees to always start at 5000; buildings at 6000, etc.

Here is a sample program to use with the Playfield Editor:

```
100 REM MAIN PROGRAM
 .
 .  OTHER STUFF AS NEEDED
 .
400 GRAPHICS 5
410 COLOR 1:X=30; Y=24; RESTORE 5000:GOSUB 1000: REM PUT THE GREEN PART
     OF A TREE AT X=30, Y=24 USING COLOR 1
420 COLOR 2: X=30; Y=24: RESTORE 5100:GOSUB 1000 : REM PUT THE BROWN PART
     OF A TREE SCENE AT THE SAME LOCATION
430 COLOR 3: X=30; Y=24: RESTORE 5200: GOSUB1000: REM PLOT THE SKY PART
     OF A TREE SCENE
440 REM REPEAT LINES 410 TO 430 USING DATA STATEMENTS FOR OTHER SHAPE
     LIKE BUILDINGS, ETC.
1000 REM *** PLOT ROUTINE ***
1010 FOR I = 1 TO 200
1020 READ A,B: IF A =999 THEN RETURN
1030 PLOT A + X, B + Y: NEXT I
1040 REM PLOTS THE SHAPE AT X,Y...PLACE THE 999,999  AT THE END OF YOUR
      DATA TO ACT AS A FLAG TO STOP READING DATA
5100 DATA 3,4,5,6,7,8,9,12,34,32,51,64,323,43,43,999,999
5200 DATA 36,56,12,5,12,78,44,32,12,34,5,67,8,999,999
5300 DATA 36,38,73,12,46,66,7,2,3,44,6,19,32,4,24,999,999
```

THE ABOVE DATA IS NOT ACTUAL DATA, BUT REPRESENTS THE FORM OF WHAT THE PROGRAM COULD LOOK LIKE. HAVE FUN WITH THE EDITOR.

A final note on this editor. It uses a custom Display List that you may find interesting to study. The method is explained in Tricky Tutorial number 1.

****

****

COLORS - EX 5.5

- NOTE -

Tape users will have this program come on the screen looking very strange. Just press reset and type run to correct the screen.

We talked about how to change the colors of the Playfield (708 to 712) a few pages ago. The following utility will allow you to pick the colors directly without having to calculate the number to poke. The pokes listed, however, are for the Player colors which live in different locations (704 to 707 + 711). Beyond this, the real purpose of this example is to introduce you to the players and missiles.

Look at the stripes of color on the screen. Each one is half a Player. The bottom half is "Turned off" so that you can see the color information.

WHAT GOOD ARE STRIPES OF COLOR?

Good question. The stripes are only being used to best show the colors of the players as you change them (I'll tell you how in a moment). You can pick which of the eight pixels (little squares of color) across each stripe that will be colored. In the next example we will select certain pixels and create a useful shape. For now the top half of each Player and missile is turned on (all 8 pixels colored- ie . poked with 1's) and the bottom half of the stripe is turned off (all eight pixels poked with 0's).

Are you wondering which Player is which? Player 0 is on the far left and Player 3 is on the right. On the far right are the corresponding missiles 0 to 3 that work with each Player. I know that calling the first Player by "#0" is confusing, but it is required by the
ATARI hardware and software.

HOW TO USE

Plug a Joystick into port 1. As soon as you move your Joystick an "*" will appear. Place the asterisk over any of the players. Now press the trigger button and move the Joystick up to increase the color value, or down to decrease it. Note that the numbers you poke are the same for any given color. This means if you find a red that you like with a value of say 45, you can poke 45 into any of the color registers from 704 to 712 and get red for the corresponding Missile, Player, or Playfield.

When you have one Player color as you want it, move the asterisk to another Player and likewise change its color. After all four players are done, you can set the background color by moving the asterisk to the far right side and pressing the trigger To make these changes permanent, you MUST ADD the pokes to your programs,

A few of you by now have tried to change the colors of the missiles. We have not yet mentioned any color registers for the missiles. This is because

there aren't any. Each missile takes on the color of the Player with the same number. Here is a review of the colors and registers:

REGISTER	USED FOR
********	********
704	Player 0
705	Player 1
706	Player 2
707	Player 3
711	Player 4

What? Player 4? Yes, you must know by now that the ATARI is always full of surprises. It turns out that the missiles can be combined together into a 5 th Player (#4). We will come back to how it is done later. For now, press SELECT. The missiles will now come together and take on the color poked into 711. To use the Color Editor on Player 4, just move the asterisk to it as before, and use the trigger. Pressing SELECT again would toggle the 5 th Player back to missiles. This would be a good utility to use anytime you want to choose a color.

****

WHICH IS THE PLAYER?

If you run the next example (5.6), you will find two robot shapes appear on your screen. You mission, if you choose to accept it, is to figure out which is a Player and which is plotted, i.e. which is drawn on a Playfield . Really try hard to see the difference (there isn't any). To solve this "Great Mystery" use a Joystick in port 1 to move each shape. The Player moves WITHOUT PRESSING THE TRIGGER. The
Playfield shape moves by PRESSING THE TRIGGER. I hate to admit it,but I cheated on this example. The Player is being moved with a small machine language utility. You will soon see that even Basic can move a Player pretty fast vertically and immediately horizontally. An interesting thing to do with this example is to change the color in the program of the Player (find a poke 704,#) and then move the Player exactly on top of the plotted shape. This will give you some ideas about how great players are. If you move two plotted shapes over each other, and then move them away, you would have to redraw them both to remove the holes left in their shapes.

This example should impress you with some ideas of shapes that can be made up by turning on parts of the Player stripe. Instead of the Robot, you could have anything you can create in the limited width of a Player (later I'll show you how to combine players for more width). Let's learn how to create a Player.

****

Press OPTION like always to get to the next example, EX 5.7. This will begin the technical part of this Tutorial

First of all, notice that the last picture is still on the screen. Do you know how this happened? It's very simple. With

playfields , when you plot a background on the screen they stay on until you erase them with either new plotted data, or usually just another Graphics call. We loaded in the next program while a Playfield (the Robot) was still on the screen, so it is still there when this example starts to run. This process is similar to adding 32 to a graphics call as explained in the Basic manual. Think of the possibilities! You can create your backgrounds with one program and then have the main loop of your program load in separately. This allows large programs to fit into smaller memory sizes. Players will stay on the screen also. In this case I changed the Player's shape.

Again, plug in Joystick 1. You will be able to move the shape on the screen in all directions, including diagonally. This routine is very important to you since it will teach you all of the following basics:

IMPORTANT STUFF TO LEARN FROM THIS EXAMPLE

1) How to create a Player shape

2) How to move that shape up and down (vertically)

3) How to move it sideways (horizontally)

This example can be copies into ANY program you want and just modified in a few places to get a Player moving around on your own design of a game or other application. It should be saved to put on a separate tape or disk and even renamed as a utility for disk users.

The complete Basic program for this example can be found in the back of this manual. I will discuss the highlights here. You may want to flip back and forth between the complete code and here as needed.

The example starts out skipping to line 140 to do some standard setups that will always be required. Line 160 you should recognize as setting the color of Player 0, which we will use for the shape (Spaceship ?) Lines 170 to 190 look like this:

```
170   PMB  =PEEK (106)-16
180 POKE 54279,   PMB
190  PMBASE =  PMB  *256
```

These lines reserve space for Player and missile shapes to be placed in. This topic alone could take pages to explain, but I will give you only enough information to make it work. See recent Magazine articles if you want the long-winded version.

Players and missiles are created by the ATARI by something called DMA. DMA means Direct Memory Access. All the big computers have it. Apples don't! This process simply means that if you have "turned on" the DMA , the computer will take whatever shapes it finds at a place in memory starting with PMBASE and put them on the screen. They will be located horizontally (X direction) by the values stores in their horizontal registers (53248 to 53255). Their size in the X direction is controlled by values stored in size registers (53256 to 53260, see below). Their size in the Y direction is controlled by two factors. First is how many pixels you have turned on for

their shape. Also, players may be created where each number in memory is put on the screen as either one pixel (single line resolution)or two (double line resolution). Their location on the screen in the Y direction is simply controlled by where in the stripe you turn on pixels . You will see all of this explained again in the examples.

WOW ! I'M TIRED!

That was a lot. There is a lot more, so take a break if you need to. Back to lines 170 to 190. Line 170 takes the value in location 106, which is the location of the top of user memory (given in number of 256 byte pages), and subtracts the specific number of pages you have predetermined. This number is then poked into 54279. That will tell the computer where the Player data starts. Finally line 190 takes PMB in number of pages and converts the value to a regular memory location, PMBASE. We will use this number a lot. The number I subtracted, 16, will change depending on the Graphics mode you are using. Choosing the amount to subtract is explained in Ex. 5.9.

Next, in line 200, we poke 53277 with a 3. This memory location is called GRACTL; poking with a 1 (Missiles only) or 3 (Players and Missiles) is one step in turning on the DMA mentioned above.

Line 210 is the initial X and Y positions where on the screen you want the Player. In 220, the size of Player 0 is set to normal, which is 0. 53256 would have contained 0 anyway, but since we are running many programs one after another, this made sure the value was correct. The choices you have for the size are:

POKE WITH	PLAYER SIZE
0	Normal size
1	Double size
3	Quadruple size

Here are the registers available for controlling Player Missile size. Poke with the appropriate values from the previous table.

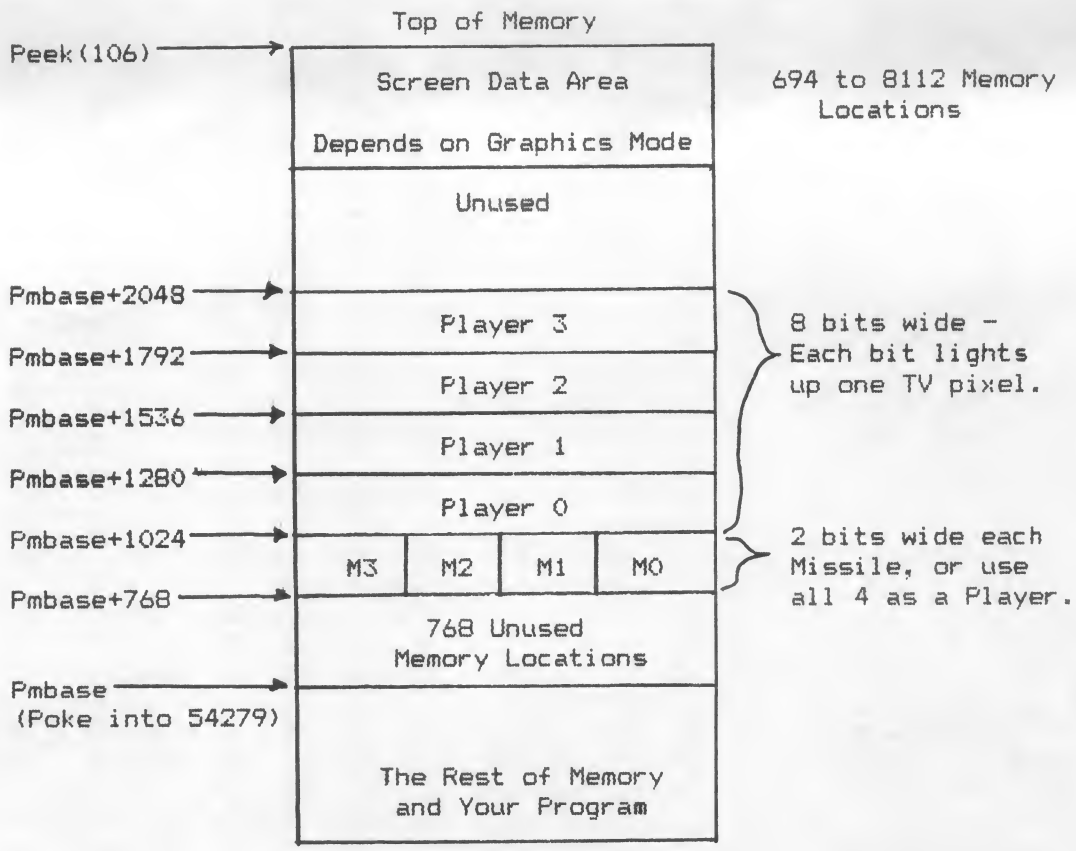
REGISTER	PARAMETER
53256	Size Player 0
53257	Size Player 1
53258	Size Player 2
53259	Size Player 3
53260	Size of all Missiles

To determine Missile size, you'll have to do a little calculating (don't worry - nothing too complicated!). Just pick the desired size for each missile from the chart below, them add up the numbers for all four (or however many Missiles you're using) and poke the total into memory location 53260. For example, if you want a normal MØ , a double M1, a quadruple M2, and a double M3, you would poke Ø + 4 + 48 + 64 = 116 into 53260. If all you wanted was a normal MØ and a quadruple M3 the number to poke would be Ø + 192 = 192.

MISSILE #	NORMAL	DOUBLE	QUAD
********	*****	******	****
0	0	1	3
1	0	4	12
2	0	16	48
3	0	64	192

Line 230 pokes 0's into the area where the data for the Player must go. This will act to clear out the players stripe first. This may not seem necessary to you. For this reason, some of the later examples will actually erase their Player areas while you are watching. You will see the junk being erased. This one statement takes a few seconds to do, and thus delays the beginning of any program that uses it. The best way to speed it up is to put it at the very front of your programs. This is demonstrated in later examples. For this example we only had to clear out memory between Pmbase + 1024 and Pmbase + 1280. Why? To explain, we need a figure of what memory looks like:

SINGLE LINE RESOLUTION



Look at the chart above marked Single Line Resolution. Starting at the place marked top of memory, we move down by a certain number of pages subtracted from the value in 106. This places us at the BOTTOM of the chart. This calculation was done in lines 170 to 190. Again, the amount to subtract will be explained in Ex. 5.9. Now that we have placed Pmbase , go back up in memory to Pmbase +768. The next 256 bytes of memory are the stripe that holds the Missiles. If any of these numbers in memory are not 0, and if the setup pokes have been made to turn on DMA , then those non-0 values will light up on the screen with the color values stored in 704. To move the shape up or down the screen, you just poke non-0 bytes up or down the stripe of memory. Let's say that you place Pmbase at 30000 in your memory. Any non-0 bytes at Pmbase +1030 (30000 +1030, or 31030) will appear near the top of the screen and if those same numbers are poked into Pmbase + 1200, they will appear near the bottom of the screen.

Continuing with this logic should make it clear what happens if you poke numbers into Pmbase + 1400. You would now be turning on a shape within Player 1 that would appear on the screen if the "setup pokes" were done. Let's finish discussing the setup procedure.

Lines 240 to 300 poke the shape into memory, just as we were talking about above. Understanding lines 250 to 300 will be a real key for you. Here is what they look like:

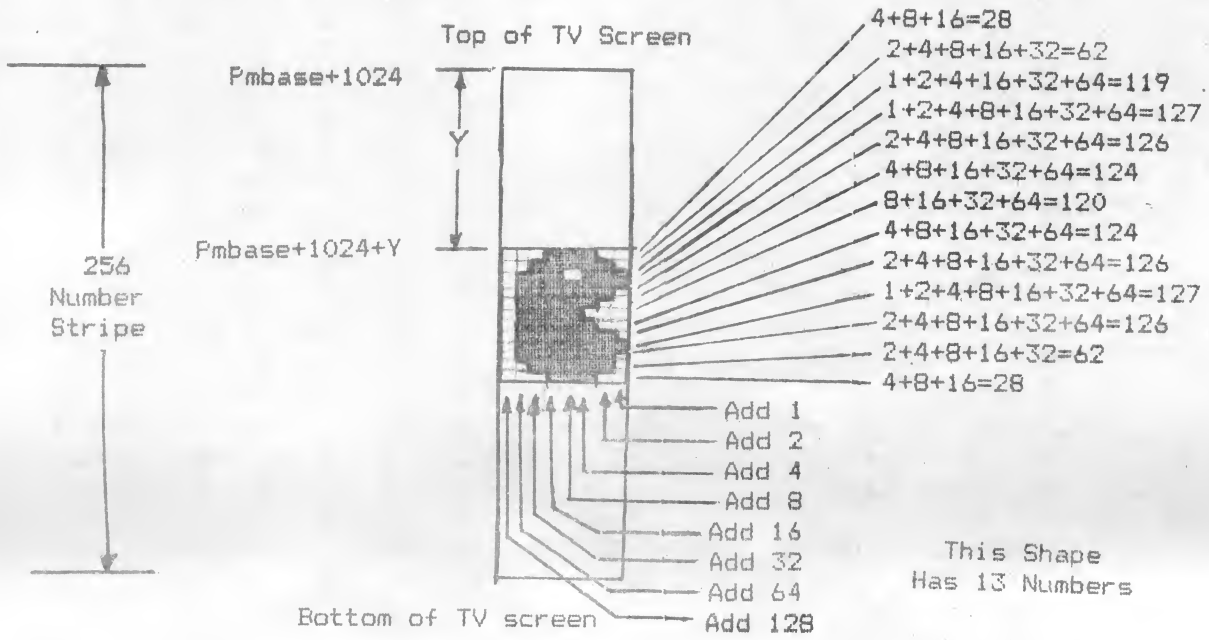
```
250 FOR I= PMBASE +1024+Y TO  PMBASE +1280
260 READ B: IF B=0 THEN 310
270 CNT=CNT+1
280 POKE I,B
290 NEXT 1
300 DATA 8,60,126,195,60,24,126,16,0,0
```

The shape is poked into memory with an offset of "Y". This offset says to place the shape Y units down the screen. If Y=0 then the shape will be at the top of the screen. A value of Y=252 puts the shape at the very bottom since the shape has eight numbers that must fit into the 256 byte long stripe of memory. In a few lines we will simply calculate where we want the shape on the screen and use this routine to poke it into the right place.

To create your own shape, you just fill in the boxes you want for the shape. Then, to get the "TOTAL", you just go across each row and add up the value for any box that is filled in. When you poke these numbers into the correct area of memory, your shape appears on the screen!

The last major hurdle is the way the data is calculated. Here is another figure to help you visualize the shape.

PLAYER 0



Back to the program code. Line 310 is what finally turns on the Player. The correct value here is obtained by adding up the options you want from the following:

DMACTL (559)

FOR	ADD	
Wide Playfield	3*	
Standard Playfield	2*	choose one only
Narrow Playfield	1*	
Turn on Missile DMA	4	
Turn on Player DMA	8	
Double Line Resolution	0*	
Single Line Resolution	16*	choose one only
Turn on Main DMA	32	

In this case we want Standard Playfield (2) + Missile (4) and Player (8) DMA plus the Main DMA (32) and Single line Resolution (16) for a total of 2+4+8+32+16=62. This is why we poke 559.62.

**********THAT ENDS THE SETUP**********

You should be able to use this same setup code for most PMG programs you do. Just make the few changes required by things like resolution and Playfield size. Note - single and double line resolution will be discussed in example 5.10 coming up soon.

The program now branches to the main program loop at line 50, which reads Joystick 0. Lines 60 and 70 increment X & Y from their initial positions set in line 210. The only thing left forthe program to do is move in the X and Y directions. The X direction is easy. The ATARI has built in position registers for the X direction. In line 110 we just poke the new X value into that memory location, 53248.

Here are the X direction registers:

REGISTER #	MISSILE/PLAYER*
53248	M0
53249	M1
53250	M2
53251	M3
53252	P0
53253	P1
53254	P2
53255	P3

* M0 refers to Missile 0, P1 refers to Player 1, etc.

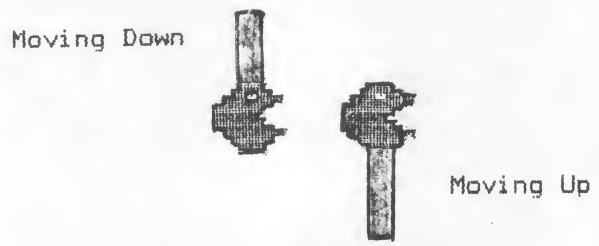
To move in the Y direction we have to do something similar to the way the Playfield Robot was moved in Ex 5.6. Fortunately, we don't have to move as many pieces of memory around. That robot had over 100 data points! Moving our Player is as simple as poking in eight numbers into memory. Look at lines 340 to 480. First, in 350, I compare the current value of Y to the last Y value. If it is the same, we don't waste our time and just return. Line 360 says, "If Y is now bigger than last time, go to the move up in memory routine at 430, otherwise move down". Remember that Y increases in memory, but appears to go down the screen. This comes from the way the ATARI does its coordinate system and is explained on page 47 of your Basic manual.

Both routines to move up or down are similar to the original one we used to poke the shape into memory. All we are doing is poking the same shape down or up by one memory location.

IMPORTANT POINT!!

When you move the shape you repoke 7 of the 8 numbers that are currently in memory and poke in 1 number in a new location. That last number is not changed because you have moved by one byte. That is why the 0 is poked into

that remaining location in line 420 or 480. Were the 0 omitted, a trail of unchanged bytes would be left on the screen, leaving something like this:



If you are lost by all this there is only one way to figure it out. Stop the lesson now, break the program, and start making small changes to where and what numbers are poked into memory. Soon you will say to yourself, "That's what he was TRYING to say!"

THAT'S IT!

You now have most of the basics to place a Player on the screen and move it around. The earlier Playfield editor discussed how to create the background for your program. Now all we need are the small details that tie all of this together. Unfortunately, there is still a lot of material to cover.

HEY! WHAT ABOUT THE PADDLES?

Oh yes, I forgot to tell you why the move in the X direction was so slow. The joysticks we are using, combined with the program code, only allow X to change by one horizontal unit at a time. Paddles, on the other hand, can instantly change the value they return from 0 to 228. Plug in a paddle controller, if you have one, and now try moving the shape across the screen. It will be instantaneous!

****

EX. 5.8 - LUNAR LANDER

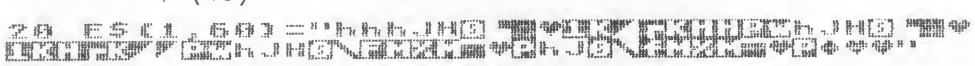
This simple example takes the same basic routine as in 5.7, but starts to make it look like a game (you may finish it if you want - good practice). There are three things you should learn from this example. First, notice the stripe of randomly flickering "junk" on the screen when the program first starts. This is caused by only partly turning on the DMA as discussed as above. As soon as all the setup is complete, the correct shape appears.

Secondly, look at the program code (in the back of this manual) and see the way the program is structured. When you write your own codes, I suggest you do something like this. The program just goes to several subroutines that:

1) Draw the mountains
2) Draw the stars
3) Set up Player 0 as in the above utility
4) Main loop routine for the "action" (movement and if desired, scoring). Note the use of sound in the loop. It's easy to do if you spend a little time to practice. This simple structuring is used in most games of the arcade type!

Finally, we offer you one of the two machine language utilities within the program. Both are short and can easily be transferred to your own programs. This one is used to move Player 0 around only. It will read Joystick 0 and move the data in the Player 0 stripe accordingly. You must use single line resolution players, as we have so far been doing (that means the shape is at PMBASE +1024 to PMBASE +1279). You should copy the needed lines to a separate file using the LIST command. You could then ENTER it as needed. Don't forget to delete the other lines of the program first. The needed lines are just 40, 90 and 130! Here is the routine:

```
10  DIM E$ (40)
```



```
130  A =  USR  (ADR (E$), STICK(0))
```

****

EX. 5.9 - A GOOFUP !

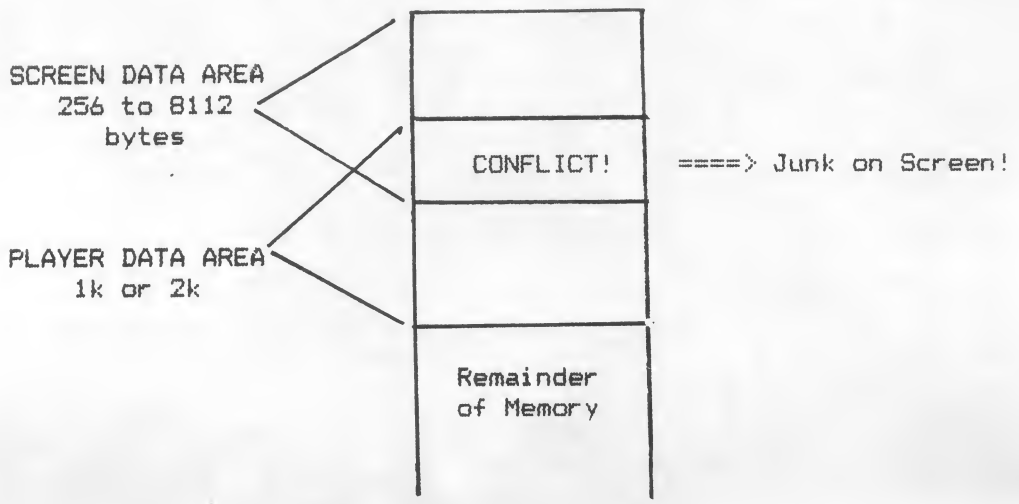
When you run this example, you will notice our old friend, the Robot, is back. However, as you move him about (using the same machine language routine as the last example), there is some "junk" on the screen. The time has come, my friends, to explain how to position the Player-missile data area within memory. Turn to the program listing for Ex. 5.9 in the back of the manual. Do you remember how we obtained the value of PMBASE ? It worked like this:

```
100 PMBASE  = (PEEK(106)-16)*256
```

If you look at page 45 of your Basic manual, you will see a chart showing the amount of RAM (memory) needed for each graphics mode. As the drawing shows, you don't want to have the data for the players and missiles overlapping the data that is placed on the screen. Even worse would be to write over the DISPLAY LIST. That is what happened to tape users with the Color example, EX 5.5. When they reran the program by pressing RESET, you fixed its Display List. In this example, you are looking at the Player data being moved by the machine language program within the Player stripe. See if you can understand the term " wraparound " by moving the Player up until it comes back from the bottom. This will also work sideways. The data that is being written to the screen area will also wrap around.

At the time I explained the method, I said that the number 16 changes, depending on your program. Let's discuss why. The listing you are looking at will only use 16 as the memory amount to subtract because the graphics mode being used requires 1k or less. Other examples, for instance Ex. 5.8 pokes memory back 40 pages. This is because that program uses Graphics 8, which requires more memory. Here is a drawing to help our discussion:

CONFLICTS
Between
PLAYER MISSILE DATA AREA
and
SCREEN DATA AREA??



The easy way to avoid putting Player data where it doesn't belong is to use the following chart. Single line resolution players always must start on a "1K boundary." This simply means subtract from the value in 106 either 8 pages, or 16, 24, 32, etc. Remember, a PAGE is 256 bytes (1/4K). When we discuss double line resolution players (soon, soon), you must start the data on a 1K boundary, so subtract 4, 8, 12, 16, etc.

GRAPHICS MODE	APPROX. # MEMORY LOCATIONS USED	SUBTRACT THIS # FROM LOCATION 106
0	992	16 (8)
1	674	16 (8)
2	424	16 (8)
3	434	16 (8)
4	694	16 (8)
5	1174	16 (8)
6	2174	16 (12)
7	4190	24 (20)
8	8112	40 (36)
9	8112	40 (36)
10	8112	40 (36)
11	8112	40 (36)

IT'S . . . DOUBLE LINE RESOLUTION (EX 5.10)

This simple example introduces you to the SQUIGLIY MONSTER. It will show itself in two sizes. The smaller size is called SINGLE LINE RESOLUTION. Simply put, each number in the data area for the player is placed on the screen as one line also. The data area has 256 bytes available, which as you saw in the Color Changing example, more than covers the screen. You can actually look closely at you TV screen and see the individual pixels.

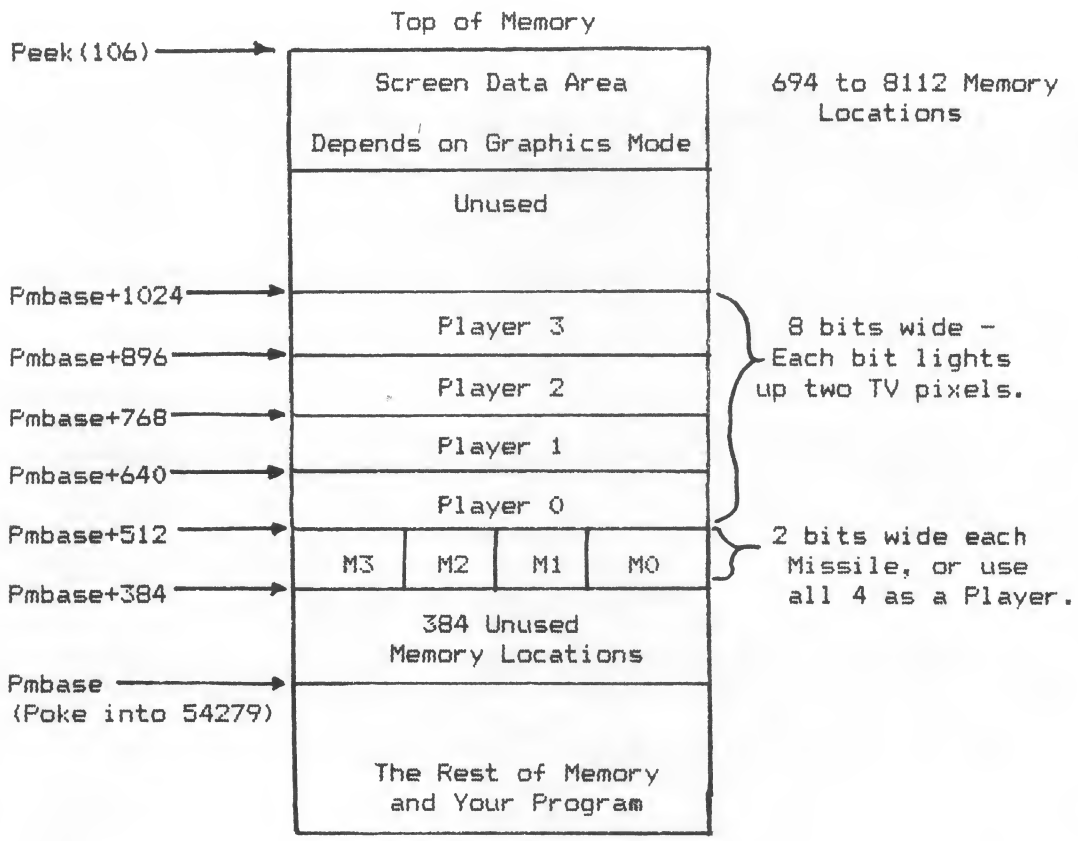
The other way you can set up your players and missiles is called DOUBLE LINE RESOLUTION. Double line uses only 128 bytes per player, so to fill the screen the computer will place TWO lines on the screen for each number in its data area. This makes the shape appear thicker; it also takes less memory for the PM data area. Press the select button for the single line player and the start button for double line.

Double line players are set up similar to single line, but with these differences:

1) Instead of adding 16 to the total value to poke into 559, add 0 (see previous chart). This tells the computer where to look for the data for each player and missile. This example switches between the two resolutions with just this one poke.

2) Instead of poking the shape of the players into Pmbase+768 to Pmbase + 2048 as the above figure showed, you place the data into Pmbase+384 to Pmbase+1024. This means that we have to look at ... ANOTHER FIGURE:

DOUBLE LINE RESOLUTION



There isn't too much more to learn before we can discuss the Pacman game (I promise to shut up in 10 or 20 pages). Next, we will animate the little Pacman shape. Run EX 5.11. You will probably see the memory area for the shape cleared out, then the shape will appear. It doesn't look much like a Pacman? No mouth? Push the joystick!

The shape should look better now that it is moving. We could have used three or four shapes to create the action of the opening and closing mouth, but two shapes is simple to explain. If you add another shape with the mouth just slightly open, and you'll see smoother action. To see the two shapes as they change, just give a series of quick pushes on the stick. What you are seeing is two players, but only one at a time. Let's see how I did this:

Look at the program code for EX 5.11. We will discuss what is being done differently than in previous examples. Notice in line 50 the memory setback is 32, not 16. This is the amount required when using Graphics mode 7. In line 60 pokes are made to both the horizontal location and size of Player 2. Yes, Player 2! You don't have to use the players in any particular order. Lines 120 to 150 just poke in the data for a second shape into the correct area for Player 2. In line 225, pokes are made to place Player 0 at X and place Player 2 at 0, which is off screen. Finally, at line 226 Player 0 is now moved off screen and Player 2 is placed at X. There is a delay on line 225 to allow you to see the "switching" of shapes.

Does that seem hard? All you have to do to create additional players is:

1) Poke into the correct memory area the data for the new player's shapes,

2) Poke the size you want into the size location in memory (optional, since 0 is a default if you forget),

3) Poke the horizontal location (X) into the horizontal memory location. If you forget, it will be off the screen to the left, at 0.

4) Have fun.

And to animate a shape you can just alternate two or more shapes at the same spot on the screen and the different shapes will seem to blend into an animated shape.

A note on animation is in order. Think back to the Robot example where I asked, "Which is the player?". I hope you can now see the power of using players for your shapes. To plot and erase these shapes from Basic would be very, very slow!

****

PRIORITY & COLLISIONS

These are two topics that were saved for last. They are not difficult subjects. I just wanted you to get a lot of experience first with the basics. EX 5.12 takes the beginning of a game that we did in EX 5.11, and adds collision with some energy pellets in order to allow the creature to "eat" the

pellets. Also, when you cross the "walls" in the tunnel, your shape will seem to be in front of the walls. At the end of the tunnel, the shape will be inside, or behind, the wall. This is all because we set the priorities so that some colors act differently than others. Here are the specifics:

Most of this program is the same as the other examples. One of the nicest things about using Players and Missiles is that you can write a program, and then have to make only slight changes to it in order to turn it into something used for a completely different purpose.

Line 10 pokes all of the horizontal location registers to 0. This is not used in this program, but makes sure that all previous players are off screen. Remember that just because we loaded in a new program, it doesn't mean that the players won't stay on the screen. They stay there until removed! This can be useful in programs that have multiple parts to them.

Lines 47 to 57 plot three different parts to the background. The first part is the "Energy Pellets", ie. small dots that the Pacman will "eat". These are plotted in Color 1 (location 708). Next, lines 50 and 52 draw out the rectangular walls that our imagination will call walls of a Pacman tunnel. The walls are drawn in Color 2. Then lines 55 to 57 draw the walls that impede the character's progress. These are a third color plotted in...you guessed it!...Color 3. The fact that different things are drawn in different colors is critical to the game. The ATARI has memory locations called COLLISION REGISTERS that can be looked at to see if something (like a player) has touched something else (like a Color 1 wall).

Next, in line 205, we learn a new trick. There is a memory location called HITCLR at 53278. It does just what it's name implies. When you poke it with a 1, it will clear the other Collision locations to read 0. If this is not done, then once a collision occurs, the appropriate collision register will be set and will remain that way. You'll see all this in action in a moment. First, lets list all the collision registers:

(NEXT PAGE)

MEMORY LOCATION	TYPE OF COLLISION
53248	Missile 0 to playfield
53249	Missile 1 to playfield
53250	Missile 2 to playfield
53251	Missile 3 to playfield
53252	Player 0 to playfield
53253	Player 1 to playfield
53254	Player 2 to playfield
53255	Player 3 to playfield
53256	Missile 0 to player
53257	Missile 1 to player
53258	Missile 2 to player
53259	Missile 3 to player
53260	Player 0 to player
53261	Player 1 to player
53262	Player 2 to player
53263	Player 3 to player
53278	Clear all collision registers

There were a lot, weren't there? They are really easy to use, however, so stay with me. You simply peek at the location that starts with the object you are concerned with. In this example, we want to know if the shape, Player 0, has collided with the playfield. That means we will look at the value in 53252. This is done in line 242. If it isn't 0, some kind of collision has occurred and we branch to the subroutine at line 1000. At 1000 to 1005, we test to see what is in the Collision register. We already knew it was not 0, since that is how we got this far. If it is 2, we know that the player "hit" (overlapped) playfield 2, which is the walls of the tunnel. We then make a collision sound and go to line 1030, which is ALWAYS REQUIRED to reset all collision registers to hold 0's. If it is a 1, we know we run into playfield 1, which in our program is the "energy pellets," so I print out the message "Yummy," and make an appropriate sound. Then it also pokes HITCLR and returns.

Does all this make sense to you? Let me repeat the basic way to use collisions: Find the Collision register that concerns the two things you are concerned with, either a Missile #_ or a Player #_ on the left side and a player, missile, or playfield number on the right. Read the value in that register by using the PEEK command. The number that results will be either 0, 1, 2, or 4. If it is 0, it means you collided with yourself which is meaningless. If it is a 1, you collided with Playfield 1, Player 1, or Missile 1, depending on the register you are using. If a 2, it means you collided with Playfield 2, Player 2, or Missile 2. A 4 means a collision with Playfield 3, Player 3, or Missile 3. You can then branch to do whatever kind of noise or graphics you want.

One last trick is in this example. Line 245 allows the character to wraparound in the X direction. It just tests for a value of X greater than

200, which is off the right side of the screen. Be setting X equal to 40, the character will suddenly appear on the left.

One part of this example doesn't show up in the code. That is priority. This term means "What will appear in front of what". In other words, will a character appear to go inside a House drawn on the screen, or pass in front of it? This example didn't poke anything into the Priority register, so the defaults were used. Here are the options for the priority register:

PRIOR (623)

For overlapping areas of players to have a third color.....Add 32

For all 4 missiles to have the color in location 711 (Playfield 3).....Add 16

For these priorities:*	Add
PF0, PF1, P0 to P3, PF2, PF3, BACKGROUND	8
PF0 to PF3, P0 to P3, BACKGROUND	4
P0 & P1, PF0 to PF3, P2 & P3, BACKGROUND	2
P0 to P3, PF0 to PF3, BACKGROUND	1

*Choose 1 only.
PF0 refers to Playfield 0.
P0 refers to Player 0.

To use this location, just add up the options and poke the TOTAL NUMBER into 623. You can only choose one set of priorities, so either choose 8, 4, 2, or 1.

A 5TH PLAYER??

Remember, in Ex. 5.5, the Color utility, how you could press SELECT and have all the missiles come together into a single player. This is how it was done: I poked 623 with 17 for both the priority I wanted and to change the colors for the fifth player (lines 1000 to 1050). Then I simply moved their X locations next to each other. Remember that missiles are only two pixels wide each. You can use them together just like a regular player. The mathematics to keep track of them will be a little harder, that's all.

****

16 PIXEL WIDE PLAYERS

You now have at least 128 pixels in the Y direction to work with. For Single line resolution you have 256. But so far we have been hindered by only 8 pixels of width. The answer is obvious, but I'll mention it anyway. Just

position two players near each other so they match up. You can write some simple math to always keep them 8 apart in the X direction. Look at Ex. 5.13 to see an ATARI logo made this way.

****

ANOTHER SURPRISE!
(32K required)

Just in case anyone has gotten to this point in the Tutorial and feels they haven't gotten their money's worth, Ex. 5.14, the last program, is an editor for you to use to create and change your Player shapes. Bill and I feel that this program alone is better than any other PMG editor currently for sale. We hope you like it! Here is how to use it:

1) When the program starts, it will request if you want to load in old data, or create new shapes. Choose NEW.

2) At this time you should get the message "Player # to edit." Pick any number 0 to 4. Player 4 is made up of the missiles.

3) A flashing cursor will appear in the editing box on the left. Press D to draw and move the cursor with the four cursor control keys (CTRL not needed).

4) To stop drawing, press E (erase). Move the cursor to the next spot you want to draw.

5) You may continue editing the shape, just press START whenever you want to see the player's shape updated.

7) Press OPTION to change the player number you are editing.

8) You may also move the player you are currently editing (with a joystick in port 1) to any place on the sreeen.

9) You can change the size of the player being edited (except number 4, sorry), pressing SELECT and choosing 0, 1, or 3 for NORMAL, DOUBLE, or QUADRUPLE size.

10) Press S to save the shapes you have made to tape or disk.

11) Press L to load any saved players back in. Disk users will find a set of players on the disk already. Remember when you load in a set of players you won't see them until you press OPTION and request each one!

This Editor should allow you to quickly create players for any purpose. Write with your own suggestions on improvements, and I will incorporate your ideas in later versions of this program. The code for this utility also is a great Tutorial on how to handle players using strings to store them in. Advanced users should study it.

****

At this point we can now see if there are any loose ideas that need to be covered. First, let's go way back to the game in Ex. 5.2. You have learned all of the technical aspects of writing a game using PMG. I want to discuss the structuring of a game. This means what goes in the program code. You should be able to figure out the rest, but if I've missed anything, please write.

Lines 40 to 60 set the colors. Line 70 points to a subroutine to draw the Playfields. Always do this towards the end of your program so that the main loop will run more quickly. The Playfield in this case consists of just plotting a lot of data. This is similar to the earlier discussion on Playfields.

Line 80 tells you that the program hasn't stopped, but is just setting up itself. This is always a nice touch for you to add.

Line 110 is tricky. I am about to present you with another machine language routine that is far more useful than the earlier one. Here I set up four starting locations that are in a safe area of memory, right between PMBASE and PMBASE+768 where the missile data starts. Since line 20 clears this whole area out, we know it is safe and available. You'll see what goes there in a moment.

Now, in lines 130 to 310, I poke in the data for two shapes for the Pacman and two for the Squigily Monster. These start at convenient increments so I can find the shapes later.

Line 330 is the machine code which is stored in a string. You may copy this line out for other programs along with its dimension statement in line 50 and its USR call. The code was written by Geoff Caras of our Group and donated to this project in exchange for "services rendered."

Lines 380 to 440 do the normal math that is needed to keep track of X and Y for TWO joysticks.

Lines 450 to 480 are the neat part. First, let me tell you what the machine language can do for you. The routine is a memory move routine. It takes the values in the memory locations you tell it, and moves them to any other location. You can use it to move an entire screen of data instantly! I use it to change the shape that is in the area where you store the two players' data. These lines simply tell the computer:

Line 450: "Moved the (Pacman) data from location 'shape 2' to the Player 0 area using 20 bytes of data." (This shape is now displayed.)

Line 460: "Same thing, but SQUIGILY data to Player 1 area." (This is also now being displayed.)

Line 470: "Move a different Pacman shape to the Player 0 area." (New Pacman shape is displayed.)

Line 480: "Move a different Squigily shape to the Player 1 area." (New Squigly shape is displayed.)

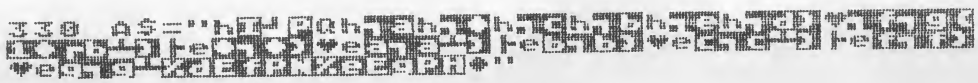
All of the above moves also take care of moving up or down the screen at the same time with the simple addition of the Y coordinate you want in the "move data to" location. This routine will move all 5 players and or/the missiles, all very quickly. The earlier machine language routine only moved Player 0. You will see this routine used a lot in our future programs.

Lines 490 to 570 take care of all the Collisions that we might be interested in, and each branches to a subroutine or makes some change on the same line.

The scoring and sounds are straight forward: Make a sound ... increment a score ... change a color. All of these things in a game depend on your imagination, i.e., what do you want to happen in the game?

Here are the specific lines to use the second machine language routine.

```
50  DIM A$(100)
```



```
450  D=USR (ADR (A$), From Address, To Address, # of bytes to move)
```

****

- FINAL NOTES -

I did not emphasize the use of missiles very much. I hope you realize that they are handled just like players. A good example of moving them is in Ex. 5.1, lines 640 to 710. Your main limits are that you only have two pixels of width, and the missile data areas are all in the same stripe. This means you must poke 0's into the data area for missiles you don't want to show on the screen, or just move them off screen with the horizontal position registers.

I left a few strange things in the programs in terms of colors, sizes, and moving shapes off screen when not in use. If you are going to understand PMG, I suggest you now take any of my examples and start to clean up little details to be more to your liking. This will be your final exam.

WILL YOU PASS??

Scores will be mailed out on Friday. BYE!!!

```
 4 REM EX5.1
 5 REM COPYRIGHT 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE
 6 REM WRITTEN BY ROBIN SHERER
 10 DIM A$(15),O$(1)
 20 GOTO 40
 30 FOR I=PMBASE+384 TO PMBASE+1024:POKE I,0:NEXT I:RETURN :REM CLEAR PM AREA
 40 K=20:REM INITIAL Y POSITION OF PLAYER 0
 50 SOUND 3,13,8,2:REM SPACE SOUND
 60 GRAPHICS 17:? #6;"  PLAYER "
 70 ? #6;"      MISSILE"
 80 ? #6;"           graphics"
 90 FOR I=1 TO 2000:NEXT I
 100 Z=90:REM INITIAL Y POS. OF PLAYER 1
 110 POKE 704,149:POKE 705,249:REM COLORS OF PLAYERS 0 & 1
 115 REM 16K USERS CHANGE P=P106-40 TO P=P106-8
 120 P106=PEEK(106):P=P106-40:POKE 54279,P:PMBASE=256*P:GOSUB 730:POKE 710,0
 130 POKE 559,46:POKE 53277,3:POKE 752,1:POKE 53248,0:POKE 53249,0
 140 GOSUB 30
 150 RESTORE 120
 160 FOR I=PMBASE+640+K TO PMBASE+644+K:READ B:POKE I,B:NEXT I
 170 DATA 153,189,255,189,153
 180 REM DATA FOR TIE SHAPE
 190 RESTORE 170
 200 FOR I=PMBASE+512+Z TO PMBASE+516+Z:READ A:POKE I,A:NEXT I
 210 POKE 53256,0:POKE 53260,18
 220 POKE 53257,0
 230 DATA 153,189,255,189,153
 240 ? "SOME CALL THIS OUTER SPACE.."
 250 FOR I=1 TO 3000:NEXT I
 260 ? "WE WILL CALL IT A PLAYFIELD"
 270 FOR I=1 TO 3000:NEXT I
 280 ? :? "    SURROUNDING THE PLAYFIELD":? "         IS A BORDER":FOR I=1 TO 20:POKE 712,RND(0)*255
 290 FOR J=1 TO 100:NEXT J:NEXT I
 300 ? :? :? "     WE CAN HAVE PLAYERS"
 310 POKE 53248,100
 320 POKE 53249,100
 330 A$="<==== PLAYER 1":X=15:Y=10
 340 GOSUB 870
 350 A$="<==== PLAYER 2":X=15:Y=150
 360 GOSUB 870
 370 FOR I=1 TO 30:NEXT I
 380 REM ******* MOVEMENT ******
 390 ? :? "  .....THEY  MOVE ...."
 400 SOUND 2,0,8,2
 410 FOR VOL=1 TO 15 STEP 0.1:SOUND 3,25,4,VOL:SOUND 1,13,4,VOL
 420 N=VOL*240/15
 430 POKE 53249,N:POKE 53248,N:NEXT VOL
 440 FOR VOL=14 TO 0 STEP -0.1:SOUND 3,25,4,VOL:SOUND 1,13,4,VOL:N=VOL*240/14:POKE 53248,N:POKE 53249,N:NEXT VOL
 450 REM ****** CHANGE SIZE *****
 460 ? :? "  ... AND CAN CHANGE SIZE TOO!"
 470 POKE 53248,74:POKE 53249,74
 480 POKE 53256,1:POKE 53257,1:FOR I=1 TO 1000:NEXT I
 490 POKE 53256,3:POKE 53257,3:FOR I=1 TO 1000:NEXT I
```

CONTINUED

```
500 ? :? " ......AND FIRE MISSILES"
510 POKE 53256,0:POKE 53257,0:GOSUB 640
516 FOR I=1 TO 8:POKE 53247+I,0:NEXT I
520 ? "}":GRAPHICS 17
530 ? #6:? #6;"      THe eNd !!"
540 ? #6:? #6:? #6:? #6;"     OF EXAMPLE 1"
550 FOR I=1 TO 1000:NEXT I
560 REM
570 REM ******* NEXT PGM **********
580 POKE 106,P106
590 ? "PRESS START TO SEE AGAIN"
600 ? "PRESS OPTION TO GO ON":POKE 53249,0
610 IF PEEK(53279)=6 THEN GOTO 40
620 IF PEEK(53279)=3 THEN POKE 764,12:RUN "D:EX5.2"
630 GOTO 610
640 REM ****** MOVE MISSILE ******
650 POKE 53252,80:POKE 53253,80
660 P=0:FOR SHOT=1 TO 6:FOR PO=20 TO 128
670 SOUND 0,PO,0,8:SOUND 1,PO,10,8:SOUND 2,PO,12,8:SOUND 3,PO,4,8
680 POKE PMBASE+383+PO,0:POKE PMBASE+384+PO,4
690 NEXT PO:NEXT SHOT
700 FOR OFF=0 TO 3:SOUND OFF,0,0,0:NEXT OFF
710 RETURN
720 REM ****** BACKGROUND   ******
730 GRAPHICS 8:CNT=0:POKE 559,0
740 COLOR 1:X=RND(0)*319:Y=RND(0)*159:CNT=CNT+1:PLOT X,Y:IF CNT>150 THEN RETURN
750 GOTO 740
760 REM ****** TITLES       ******
770 GRAPHICS 17
780 ? #6;"      PROGRAM      "
790 ? #6;"       # 1 OF      "
800 ? #6:? #6:? #6
810 ? #6;"PLAYER"
820 ? #6;"     missile"
830 ? #6;"           graphics"
840 FOR I=1 TO 50:FOR J=1 TO 5:POKE 708+J,RND(0)*200:NEXT J:NEXT I
850 RETURN
860 REM ****** MODE8 TEXT   ******
870 I1=PEEK(88)+PEEK(89)*256
880 I2=I1+Y*40+X
890 FOR Z=1 TO LEN(A$)
900 O$=A$(Z,Z):GOSUB 950
910 I3=57344+X*8
920 FOR U=0 TO 7
930 POKE I2+U*40,PEEK(I3+U)
940 NEXT U:I2=I2+1:NEXT Z
950 X=ASC(O$):IF X>127 THEN X=X-128
960 IF X>31 AND X<96 THEN X=X-32:RETURN
970 IF X<32 THEN X=X+64
980 RETURN
990 REM ******* COLLISION **********
1000 REM *** WANT TO SEE AGAIN? ***
1010 REM ** POKE 106 BACK ****
```

— EX5.2 —

```
5 REM COPYRIGHT 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE. PACMAN CHARACTER COPYRIGHT ATARI INC.
10 GOTO 580
20 FOR I=PMBASE TO PMBASE+2048:POKE I,0:NEXT I:RETURN
30 REM *****PLAYER SETUP *****
40 GRAPHICS 7:POKE 710,121:POKE 712,0:POKE 704,41:POKE 705,41:POKE 559,62
50 DIM A$(100):POKE 623,1
60 POKE 708,103:POKE 752,1:POKE 709,148
70 GOSUB 690:REM DRAW BACKGROUND
80 ? " *** WAIT  A  MOMENT  PLEASE ***"
90 A=PEEK(106)-32:POKE 54279,A:PMBASE=A*256
100 POKE 53277,3:POKE 53256,0:POKE 53250,150
110 GOSUB 20:SHAPE1=PMBASE:SHAPE2=PMBASE+50:SHAPE3=PMBASE+100:SHAPE4=PMBASE+150
120 RESTORE 160:Y1=172:Y2=172:X1=60:X2=190
130 FOR I=PMBASE TO PMBASE+20:READ B
140 POKE I,B:NEXT I
150 REM ***** DATA FOR PACMAN1 *****
160 DATA 0,0,0,28,62,119,127,126,124,120,124,126,127,126,62,28,0,0,0,0,0
170 RESTORE 210
180 FOR I=PMBASE+50 TO PMBASE+70:READ B
190 POKE I,B:NEXT I
200 REM ***** DATA FOR PACMAN2 ******
210 DATA 0,0,0,28,62,118,126,126,126,126,127,125,126,62,28,0,0,0,0,0
220 POKE 559,62
230 RESTORE 260:FOR I=PMBASE+100 TO PMBASE+120:READ B
240 POKE I,B:NEXT I
250 REM ***** DATA FOR MONSTER1 *****
260 DATA 0,0,0,0,124,214,214,254,124,68,68,204,0,0,0,0,0,0,0,0,0
270 RESTORE 310
280 FOR I=PMBASE+150 TO PMBASE+170:READ B
290 POKE I,B:NEXT I
300 REM ***** DATA FOR MONSTER2 ********
310 DATA 0,0,0,0,124,254,214,214,124,68,68,102,0,0,0,0,0,0,0,0,0,0
320 SCOREMON=3:SCOREPAC=3
330 A$="hIPQhah'hchbhehd)fg(1'b)e'')eaa)ebb)ecc)eff)egg%dE{PN%eEgPH'"
340 ? :? :? :? "STICK1                STICK2":FOR I=1 TO 300:NEXT I
350 REM ******MAIN LOOP ******
360 GOSUB 1030
370 POKE 53278,1:REM CLEAR ALL COLISION LOCATIONS (SO THEY HOLD 0'S)
380 ST1=STICK(0):S=PEEK(53279)
390 ST2=STICK(1)
400 IF S=3 THEN POKE 764,12:RUN "D:EX5.3"
410 Y1=Y1+(ST1=13)*2-(ST1=14)*2
420 Y2=Y2+(ST2=13)*2-(ST2=14)*2
430 X1=X1+(ST1=7)-(ST1=11):POKE 53248,X1
440 X2=X2+(ST2=7)-(ST2=11):POKE 53249,X2
450 D=USR(ADR(A$),SHAPE2,PMBASE+1024+Y1,20)
460 D=USR(ADR(A$),SHAPE4,PMBASE+1280+Y2,20)
470 D=USR(ADR(A$),SHAPE1,PMBASE+1024+Y1,20)
480 D=USR(ADR(A$),SHAPE3,PMBASE+1280+Y2,20)
```

CONTINUED

```
490 IF PEEK(53252)=2 THEN POKE 704,200:POKE 705,41:SOUND 0,200,10,8:FOR I=1 TO 30:NEXT I:SOUND 0,0,0,0
500 IF PEEK(53252)=1 THEN X1=X1-(ST1=7)+(ST1=11):POKE 53248,X1:Y1=Y1-(ST1=13)*3+(ST1=14)*3
510 IF PEEK(53253)=2 THEN POKE 705,200:POKE 704,41:SOUND 0,200,10,8:FOR I=1 TO 30:NEXT I:SOUND 0,0,0,0
520 IF PEEK(53253)=1 THEN X2=X2-(ST2=7)+(ST2=11):POKE 53249,X2:Y2=Y2-(ST2=13)*3+(ST2=14)*3
530 IF PEEK(53260)<>0 THEN GOSUB 920:GOTO 570
540 IF PEEK(53253)=4 THEN GOSUB 890
550 IF RND(0)*100>99 THEN COLOR 3:PLOT 35,55:PLOT 135,65
560 IF PEEK(53252)=4 THEN GOSUB 860
570 POKE 53278,0:GOTO 380
580 GRAPHICS 17
590 ? #6;"WE ARE GOING TO PLAY"
600 ? #6;"WITH A COMPLETE GAME"
610 ? #6;" FIRST TO STUDY THE "
620 ? #6;"  FEATURES THAT THE "
630 ? #6;"ATARI(TM) OFFERS US."
640 ? #6;"THEN WE WILL TEACH  "
650 ? #6;"YOU HOW EACH PART OF"
660 ? #6;"THE GAME WAS DONE..."
670 ? #6;"PRESENTING...PACMAN?"
680 FOR I=1 TO 3000:NEXT I:GOTO 30
690 REM **** BACKGROUND **********************
700 RESTORE 740:COLOR 1
710 READ X1,Y1,X2,Y2:IF X1=999 THEN 820
720 PLOT X1,Y1:DRAWTO X2,Y2
730 GOTO 710
740 DATA 20,20,20,0,20,0,0,0,0,0,0,79,0,79,70,79,70,79,70,70,0,50,10,50,0,40,10,40,10,10,10,30,10,30,20,30
750 DATA 20,30,20,40,20,40,40,40,40,40,40,10,20,50,20,60,20,60,10,60,10,60,10,70,10,70,50,70,50,70,50,40
760 DATA 30,30,30,0,30,0,70,0,70,0,70,10,50,0,50,30,50,40,60,40,60,10,60,50,30,50,40,50,30,60,40,60
770 DATA 70,30,90,30,90,30,90,50,70,50,70,30,70,20,100,20,100,20,100,60,90,10,90,20,80,10,80,0
780 DATA 80,0,159,0,159,0,159,70,159,79,80,79,80,79,80,60,80,60,60,60,60,60,60,70
790 DATA 90,60,90,70,90,70,150,70,150,70,150,40,140,60,130,60,130,60,130,70,130,40,110,40,110,40,110,70
800 DATA 100,30,130,30,110,10,110,20,110,20,150,20,140,20,140,50,140,50,120,50,120,50,120,60
810 DATA 150,10,150,30,140,0,140,10,130,10,130,20,120,0,120,10,100,0,100,10,999,99,9,9
820 REM ******* FILL BOXES ********
830 COLOR 2:FOR I=1 TO 20:PLOT 70,30+I:DRAWTO 90,30+I:NEXT I
840 COLOR 3
850 RETURN
860 REM ***** SCORE FOR PAC *************
870 COLOR 0:PLOT 35,55:PLOT 135,65:SCOREPAC=SCOREPAC+1:FOR I=50 TO 1 STEP -1:SOUND 0,I,10,8:NEXT I
880 GOSUB 1030:RETURN
890 REM *** SCORE MONSTER *******
900 COLOR 0:PLOT 35,55:PLOT 135,65:SCOREMON=SCOREMON+1:FOR I=50 TO 1 STEP -1:SOUND 0,I,10,8:NEXT I
910 GOSUB 1030:RETURN
920 REM **** PAC HITS MONSTER ************
930 IF PEEK(53260)=2 AND PEEK(704)=200 THEN SCOREPAC=SCOREPAC+5:GOSUB 1030:GOTO 950
940 GOTO 960
950 FOR I=200 TO 0 STEP -1:SOUND 0,I,10,8:NEXT I
960 REM **** MON HITS PACMAN ******
970 IF PEEK(53261)=1 AND PEEK(705)=200 THEN SCOREMON=SCOREMON+5:GOSUB 1030:GOTO 990
980 GOTO 1000
990 FOR I=0 TO 200 STEP 1:SOUND 0,I,10,8:NEXT I:SOUND 0,0,0,0
1000 X1=60:X2=190:Y1=172:Y2=172
1010 DUM=USR(ADR(A$),PMBASE+200,PMBASE+1024+Y,240)
1020 DUM=USR(ADR(A$),PMBASE+200,PMBASE+1024,510)
1030 POKE 656,1:POKE 657,2:? "PACMAN= ";SCOREPAC;"          SQUIGILY= ";SCOREMON:RETURN
```

```
5 REM COPYRIGHT 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE. PACMAN CHARACTER COPYRIGHT ATARI INC.
10 GOTO 580
20 FOR I=PMBASE TO PMBASE+2048:POKE I,0:NEXT I:RETURN
30 REM *****PLAYER SETUP *****
40 GRAPHICS 7:POKE 710,121:POKE 712,0:POKE 704,41:POKE 705,41:POKE 559,62
50 DIM A$(100):POKE 623,1
60 POKE 708,103:POKE 752,1:POKE 709,148
70 GOSUB 690:REM DRAW BACKGROUND
80 ? " *** WAIT  A  MOMENT  PLEASE ***"
90 A=PEEK(106)-32:POKE 54279,A:PMBASE=A*256
100 POKE 53277,3:POKE 53256,0:POKE 53250,150
110 GOSUB 20:SHAPE1=PMBASE:SHAPE2=PMBASE+50:SHAPE3=PMBASE+100:SHAPE4=PMBASE+150
120 RESTORE 160:Y1=172:Y2=172:X1=60:X2=190
130 FOR I=PMBASE TO PMBASE+20:READ B
140 POKE I,B:NEXT I
150 REM ***** DATA FOR PACMAN1 *****
160 DATA 0,0,0,28,62,119,127,126,124,120,124,126,127,126,62,28,0,0,0,0,0
170 RESTORE 210
180 FOR I=PMBASE+50 TO PMBASE+70:READ B
190 POKE I,B:NEXT I
200 REM ***** DATA FOR PACMAN2 ******
210 DATA 0,0,0,28,62,118,126,126,126,126,127,125,126,62,28,0,0,0,0,0
220 POKE 559,62
230 RESTORE 260:FOR I=PMBASE+100 TO PMBASE+120:READ B
240 POKE I,B:NEXT I
250 REM ***** DATA FOR MONSTER1 *****
260 DATA 0,0,0,0,124,214,214,254,124,68,68,204,0,0,0,0,0,0,0,0,0
270 RESTORE 310
280 FOR I=PMBASE+150 TO PMBASE+170:READ B
290 POKE I,B:NEXT I
300 REM ***** DATA FOR MONSTER2 ********
310 DATA 0,0,0,0,124,254,214,214,124,68,68,102,0,0,0,0,0,0,0,0,0
320 SCOREMON=3:SCOREPAC=3
330 A$="hIP@hah'hchbhehd)fg(1'b)e'')eaa)ebb)ecc)eff)egg%dEfPN%eEgPH'"
340 ? :? :? :? "STICK1                STICK2":FOR I=1 TO 300:NEXT I
350 REM ******MAIN LOOP ******
360 GOSUB 1030
370 POKE 53278,1:REM CLEAR ALL COLISION LOCATIONS (SO THEY HOLD 0'S)
380 ST1=STICK(0):S=PEEK(53279)
390 ST2=STICK(1)
400 IF S=3 THEN POKE 764,12:RUN "D:EX5.3"
410 Y1=Y1+(ST1=13)*2-(ST1=14)*2
420 Y2=Y2+(ST2=13)*2-(ST2=14)*2
430 X1=X1+(ST1=7)-(ST1=11):POKE 53248,X1
440 X2=X2+(ST2=7)-(ST2=11):POKE 53249,X2
450 D=USR(ADR(A$),SHAPE2,PMBASE+1024+Y1,20)
460 D=USR(ADR(A$),SHAPE4,PMBASE+1280+Y2,20)
470 D=USR(ADR(A$),SHAPE1,PMBASE+1024+Y1,20)
480 D=USR(ADR(A$),SHAPE3,PMBASE+1280+Y2,20)
```

CONTINUED

```
490 IF PEEK(53252)=2 THEN POKE 704,200:POKE 705,41:SOUND 0,200,10,8:FOR I=1 TO 30:NEXT I:SOUND 0,0,0,0
500 IF PEEK(53252)=1 THEN X1=X1-(ST1=7)+(ST1=11):POKE 53248,X1:Y1=Y1-(ST1=13)*3+(ST1=14)*3
510 IF PEEK(53253)=2 THEN POKE 705,200:POKE 704,41:SOUND 0,200,10,8:FOR I=1 TO 30:NEXT I:SOUND 0,0,0,0
520 IF PEEK(53253)=1 THEN X2=X2-(ST2=7)+(ST2=11):POKE 53249,X2:Y2=Y2-(ST2=13)*3+(ST2=14)*3
530 IF PEEK(53260)<>0 THEN GOSUB 920:GOTO 570
540 IF PEEK(53253)=4 THEN GOSUB 890
550 IF RND(0)*100>99 THEN COLOR 3:PLOT 35,55:PLOT 135,65
560 IF PEEK(53252)=4 THEN GOSUB 860
570 POKE 53278,0:GOTO 380
580 GRAPHICS 17
590 ? #6;"WE ARE GOING TO PLAY"
600 ? #6;"WITH A COMPLETE GAME"
610 ? #6;" FIRST TO STUDY THE "
620 ? #6;"  FEATURES THAT THE "
630 ? #6;"ATARI(TM) OFFERS US."
640 ? #6;"THEN WE WILL TEACH  "
650 ? #6;"YOU HOW EACH PART OF"
660 ? #6;"THE GAME WAS DONE..."
670 ? #6;"PRESENTING...PACMAN?"
680 FOR I=1 TO 3000:NEXT I:GOTO 30
690 REM **** BACKGROUND **********************
700 RESTORE 740:COLOR 1
710 READ X1,Y1,X2,Y2:IF X1=999 THEN 820
720 PLOT X1,Y1:DRAWTO X2,Y2
730 GOTO 710
740 DATA 20,20,20,0,20,0,0,0,0,0,0,79,0,79,70,79,70,79,70,70,0,50,10,50,0,40,10,40,10,10,10,30,10,30,20,30
750 DATA 20,30,20,40,20,40,40,40,40,40,40,10,20,50,20,60,20,60,10,60,10,60,10,70,10,70,50,70,50,70,50,40
760 DATA 30,30,30,0,30,0,70,0,70,0,70,10,50,0,50,30,50,40,60,40,60,10,60,50,30,50,40,50,30,60,40,60
770 DATA 70,30,90,30,90,30,90,50,70,50,70,30,70,20,100,20,100,20,100,60,90,10,90,20,80,10,80,0
780 DATA 80,0,159,0,159,0,159,70,159,79,80,79,80,79,80,60,80,60,60,60,60,60,60,70
790 DATA 90,60,90,70,90,70,150,70,150,70,150,40,140,60,130,60,130,60,130,70,130,40,110,40,110,40,110,70
800 DATA 100,30,130,30,110,10,110,20,110,20,150,20,140,20,140,50,140,50,120,50,120,50,120,60
810 DATA 150,10,150,30,140,0,140,10,130,10,130,20,120,0,120,10,100,0,100,10,999,99,9,9
820 REM ******** FILL BOXES *********
830 COLOR 2:FOR I=1 TO 20:PLOT 70,30+I:DRAWTO 90,30+I:NEXT I
840 COLOR 3
850 RETURN
860 REM ***** SCORE FOR PAC *************
870 COLOR 0:PLOT 35,55:PLOT 135,65:SCOREPAC=SCOREPAC+1:FOR I=50 TO 1 STEP -1:SOUND 0,I,10,8:NEXT I
880 GOSUB 1030:RETURN
890 REM *** SCORE MONSTER ********
900 COLOR 0:PLOT 35,55:PLOT 135,65:SCOREMON=SCOREMON+1:FOR I=50 TO 1 STEP -1:SOUND 0,I,10,8:NEXT I
910 GOSUB 1030:RETURN
920 REM **** PAC HITS MONSTER ************
930 IF PEEK(53260)=2 AND PEEK(704)=200 THEN SCOREPAC=SCOREPAC+5:GOSUB 1030:GOTO 950
940 GOTO 960
950 FOR I=200 TO 0 STEP -1:SOUND 0,I,10,8:NEXT I
960 REM **** MON HITS PACMAN *******
970 IF PEEK(53261)=1 AND PEEK(705)=200 THEN SCOREMON=SCOREMON+5:GOSUB 1030:GOTO 990
980 GOTO 1000
990 FOR I=0 TO 200 STEP 1:SOUND 0,I,10,8:NEXT I:SOUND 0,0,0,0
1000 X1=60:X2=190:Y1=172:Y2=172
1010 DUM=USR(ADR(A$),PMBASE+200,PMBASE+1024+Y,240)
1020 DUM=USR(ADR(A$),PMBASE+200,PMBASE+1024,510)
1030 POKE 656,1:POKE 657,2:? "PACMAN= ";SCOREPAC;"          SQUIGILY= ";SCOREMON:RETURN
```

```
 150 IF D=62 THEN GOSUB 3000
 152 IF D=0 THEN GOSUB 4000:REM LOAD
 154 IF D=50 THEN QQ=0:NN=1:QQ$="D":NN$="A":POKE 764,255:GOTO 170
 156 IF D=31 THEN QQ=1:NN=0:QQ$="A":NN$="D":POKE 764,255:GOTO 170
 158 IF D=30 THEN QQ=2:NN=0:QQ$="B":NN$="D":POKE 764,255:GOTO 170
 159 IF D=26 THEN QQ=3:NN=0:QQ$="C":NN$="D":POKE 764,255:GOTO 170
 170 ON YY GOSUB 171,172,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190
 171 R0$(XX-30,XX-30)=QQ$:GOTO K2
 172 R1$(XX-30,XX-30)=QQ$:GOTO K2
 173 R2$(XX-30,XX-30)=QQ$:GOTO K2
 174 R3$(XX-30,XX-30)=QQ$:GOTO K2
 175 R4$(XX-30,XX-30)=QQ$:GOTO K2
 176 R5$(XX-30,XX-30)=QQ$:GOTO K2
 177 R6$(XX-30,XX-30)=QQ$:GOTO K2
 178 R7$(XX-30,XX-30)=QQ$:GOTO K2
 179 R8$(XX-30,XX-30)=QQ$:GOTO K2
 180 R9$(XX-30,XX-30)=QQ$:GOTO K2
 181 R10$(XX-30,XX-30)=QQ$:GOTO K2
 182 R11$(XX-30,XX-30)=QQ$:GOTO K2
 183 R12$(XX-30,XX-30)=QQ$:GOTO K2
 184 R13$(XX-30,XX-30)=QQ$:GOTO K2
 185 R14$(XX-30,XX-30)=QQ$:GOTO K2
 186 R15$(XX-30,XX-30)=QQ$:GOTO K2
 187 R16$(XX-30,XX-30)=QQ$:GOTO K2
 188 R17$(XX-30,XX-30)=QQ$:GOTO K2
 189 R18$(XX-30,XX-30)=QQ$:GOTO K2
 190 R19$(XX-30,XX-30)=QQ$:GOTO K2
 192 POKE 764,255:GOTO 130
 660 REM #### DATA STATEMENTS ####
 665 FOR PFC=0 TO 2
 670 R=1:GOSUB 12
 680 POKE 87,0:POKE 88,P882:POKE 89,P892
 690 POSITION 0,0:? #6;"       PLAYFIELD #   DATA STATEMENTS"
 700 POSITION 18,0:? #6;PFC+1
 710 GOSUB 1030
 720 FOR I=0 TO CNT(PFC)
 730 IF I=20 THEN R=R+4:GOSUB 1030
 740 IF I=40 THEN R=R+4:GOSUB 1030
 750 IF I=60 THEN R=1:GOSUB 1090
 752 IF I=80 THEN R=R+4:GOSUB 1030
 753 IF CNT(PFC)=0 THEN 770
 754 IF I=100 THEN R=R+4:GOSUB 1030
 755 IF I=120 THEN R=1:GOSUB 5000:GOTO 770
 756 IF CNT(PFC)=0 THEN 770
 757 IF I=0 OR I=20 OR I=40 OR I=60 THEN GOTO 760
 758 ? #6;",";X(PFC,I);",";Y(PFC,I);
 759 GOTO 770
 760 ? #6;X(PFC,I);",";Y(PFC,I);
 770 NEXT I
 775 IF PEEK(53279)<>6 THEN 775
 777 NEXT PFC
 780 GOSUB 13:POKE 87,5:POKE 88,P88:POKE 89,P89:GOSUB 82:RETURN
 790 REM ##### ERASE SHAPE ####
```

CONTINUED

```
800 COLOR 0
810 FOR Y=1 TO 20:FOR X=31 TO 50:PLOT X,Y:NEXT X:NEXT Y
820 RETURN
830 REM ****** SET UP MIXED MODE ******
840 P88=PEEK(88):P89=PEEK(89)
850 DL=PEEK(560)+256*PEEK(561)
860 Z1=PEEK(DL+4):Z2=PEEK(DL+5)
870 I=1:MODE1=Z1+256*Z2
880 MODE2=500+MODE1
900 P892=INT(MODE2/256):P882=MODE2-P892*256
910 RESTORE 1000
930 READ S:IF S=0 THEN RETURN
940 POKE (DL+I-1),S
950 IF I=5 THEN POKE DL+4,Z1
960 IF I=6 THEN POKE DL+5,Z2
970 I=I+1:GOTO 930
1000 DATA 112,112,112,74,64,156,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
1010 DATA 2,2,2,2,2,2,2,2,2,2,2,2,2,65,32,124,0,0,0
1020 GOTO 240
1030 POSITION 0,R:? #6;LINENO;" DATA ";
1040 LINENO=LINENO+10
1050 RETURN
1090 IF PEEK(53279)<>6 THEN 1090
1091 HLDI=I:GOSUB 13:I=HLDI
1092 GOSUB 1030:RETURN
1100 FOR I=1 TO 3:FOR J=1 TO 20:? X(I,J),Y(I,J);:NEXT J:NEXT I
1110 RETURN
1200 REM
2000 REM CLEAR DRAWING BOARD
2011 GOSUB 5
2012 FOR I=1 TO 20
2014 POSITION 31,I
2016 ? #6;R0$(1,20)
2018 NEXT I
2030 RETURN
3000 REM OUTPUT DATA TO DISK OR TAPE
3001 ROW=0:XR=31:YR=1:TRAP 10000
3002 POKE 87,0:POKE 88,P882:POKE 89,P892
3004 POSITION 0,10:? #6;"OUTPUT TO TAPE (T) OR DISK (D)?"
3010 D=PEEK(764)
3040 IF D=45 THEN GOSUB 3200:RETURN
3044 IF D=58 THEN GOSUB 3300:RETURN
3048 GOTO 3010
3200 REM WRITE DATA TO TAPE
3201 GOSUB 13:GOSUB 82
3222 FOR I=1 TO 128:P$(LEN(P$)+1)=" ":NEXT I
3224 W$(1,20)=R0$:W$(21,40)=R1$:W$(41,60)=R2$:W$(61,80)=R3$:W$(81,100)=R4$
3226 X$(1,20)=R5$:X$(21,40)=R6$:X$(41,60)=R7$:X$(61,80)=R8$:X$(81,100)=R9$
3228 Y$(1,20)=R10$:Y$(21,40)=R11$:Y$(41,60)=R12$:Y$(61,80)=R13$:Y$(81,100)=R14$
3229 Z$(1,20)=R15$:Z$(21,40)=R16$:Z$(41,60)=R17$:Z$(61,80)=R18$:Z$(81,100)=R19$
3230 TRAP OPN1:OPEN #1,8,0,"C:"
3250 TRAP RAW:? #1;P$:P$=""
3260 ? #1;W$
3270 ? #1;X$
3280 ? #1;Y$
3290 ? #1;Z$
```

CONTINUED

```
3291 TRAP CLS1:CLOSE #1
3292 TRAP MLB:POKE 87,5:POKE 88,P88:POKE 89,P89
3298 RETURN
3300 REM WRITE DATA TO DISK
3301 GOSUB 13:GOSUB 82
3314 W$(1,20)=R0$:W$(21,40)=R1$:W$(41,60)=R2$:W$(61,80)=R3$:W$(81,100)=R4$
3316 X$(1,20)=R5$:X$(21,40)=R6$:X$(41,60)=R7$:X$(61,80)=R8$:X$(81,100)=R9$
3318 Y$(1,20)=R10$:Y$(21,40)=R11$:Y$(41,60)=R12$:Y$(61,80)=R13$:Y$(81,100)=R14$
3320 Z$(1,20)=R15$:Z$(21,40)=R16$:Z$(41,60)=R17$:Z$(61,80)=R18$:Z$(81,100)=R19$
3330 TRAP OPN1:OPEN #1,8,0,"D:PF1SDATA"
3360 TRAP RAW:? #1;W$
3370 ? #1;X$
3380 ? #1;Y$
3390 ? #1;Z$
3391 TRAP CLS1:CLOSE #1
3392 TRAP MLB:POKE 87,5:POKE 88,P88:POKE 89,P89
3398 RETURN
4000 REM INPUT DATA FROM TAPE OR DISK
4001 ROW=0:YR=1:XR=31:TRAP 10000
4002 POKE 87,0:POKE 88,P882:POKE 89,P892
4004 POSITION 0,10:? #6;"INPUT FROM TAPE (T) OR DISK (D)?"
4006 POKE 764,255
4010 D=PEEK(764)
4020 IF D=45 THEN GOSUB 4200:RETURN
4030 IF D=58 THEN GOSUB 4300:RETURN
4040 GOTO 4010
4200 REM INPUT FROM TAPE
4201 GOSUB 13:GOSUB 82
4204 TRAP OPN1:OPEN #1,4,0,"C:"
4208 TRAP RAW:INPUT #1,P$:P$=""
4210 INPUT #1,W$
4220 INPUT #1,X$
4230 INPUT #1,Y$
4240 INPUT #1,Z$:TRAP CLS1:CLOSE #1
4242 TRAP MLB:POKE 87,5:POKE 88,P88:POKE 89,P89
4246 VNR=20
4250 FOR NR=1 TO 5
4252 GOSUB 6000
4254 POSITION XR,YR+NR-1
4258 PRINT #6;W$((NR*VNR)-(VNR-1),NR*20)
4259 NEXT NR
4260 YR=YR+NR-1
4262 FOR NR=1 TO 5
4263 GOSUB 6000
4264 POSITION XR,YR+NR-1
4268 PRINT #6;X$((NR*VNR)-(VNR-1),NR*20)
4269 NEXT NR
4270 YR=YR+NR-1
4272 FOR NR=1 TO 5
4273 GOSUB 6000
4274 POSITION XR,YR+NR-1
4278 PRINT #6;Y$((NR*VNR)-(VNR-1),NR*20)
4279 NEXT NR
4280 YR=YR+NR-1
4282 FOR NR=1 TO 5
4283 GOSUB 6000
4284 POSITION XR,YR+NR-1
4288 PRINT #6;Z$((NR*VNR)-(VNR-1),NR*20)
4289 NEXT NR
4290 RETURN
```


CONTINUED

```
4300 REM INPUT FROM DISK
4301 GOSUB 13:GOSUB 82
4304 TRAP OPN1:OPEN #1,4,0,"D:PF1SDATA"
4310 TRAP RAW:INPUT #1,W$
4320 INPUT #1,X$
4330 INPUT #1,Y$
4340 INPUT #1,Z$:TRAP CLS1:CLOSE #1
4342 TRAP MLB:POKE 87,5:POKE 88,P88:POKE 89,P89
4346 VNR=20
4350 FOR NR=1 TO 5
4352 GOSUB 6000
4354 POSITION XR,YR+NR-1
4358 PRINT #6;W$((NR*VNR)-(VNR-1),NR*20)
4359 NEXT NR
4360 YR=YR+5
4362 FOR NR=1 TO 5
4363 GOSUB 6000
4364 POSITION XR,YR+NR-1
4368 PRINT #6;X$((NR*VNR)-(VNR-1),NR*20)
4369 NEXT NR
4370 YR=YR+5
4372 FOR NR=1 TO 5
4373 GOSUB 6000
4374 POSITION XR,YR+NR-1
4378 PRINT #6;Y$((NR*VNR)-(VNR-1),NR*20)
4379 NEXT NR
4380 YR=YR+5
4382 FOR NR=1 TO 5
4383 GOSUB 6000
4384 POSITION XR,YR+NR-1
4388 PRINT #6;Z$((NR*VNR)-(VNR-1),NR*20)
4389 NEXT NR
4390 RETURN
5000 GOSUB 13:? "TOO MUCH DATA":I=CNT(PFC):RETURN
6000 REM REBUILD STRING MATRIX
6008 ROW=ROW+1:V=6029
6009 ON ROW GOTO 6010,6011,6012,6013,6014,6015,6016,6017,6018,6019,6020,6021,6022,6023,6024,6025,6026,6027,6028,V
6010 R0$=W$(1,20):RETURN
6011 R1$=W$(21,40):RETURN
6012 R2$=W$(41,60):RETURN
6013 R3$=W$(61,80):RETURN
6014 R4$=W$(81,100):RETURN
6015 R5$=X$(1,20):RETURN
6016 R6$=X$(21,40):RETURN
6017 R7$=X$(41,60):RETURN
6018 R8$=X$(61,80):RETURN
6019 R9$=X$(81,100):RETURN
6020 R10$=Y$(1,20):RETURN
6021 R11$=Y$(21,40):RETURN
6022 R12$=Y$(41,60):RETURN
6023 R13$=Y$(61,80):RETURN
6024 R14$=Y$(81,100):RETURN
6025 R15$=Z$(1,20):RETURN
6026 R16$=Z$(21,40):RETURN
6027 R17$=Z$(41,60):RETURN
6028 R18$=Z$(61,80):RETURN
6029 R19$=Z$(81,100):RETURN
```

CONTINUED

```
 7000 POKE 87,0:POKE 88,P882:POKE 89,P892:POSITION 0,0
 7004 ? #6;"WHATEVER YOU DID REALLY BEFUDDLED ME  "
 7006 ? #6;"MUCH AS I HATE TO ADMIT CONFUSION  "
 7007 ? #6;"WE'RE JUST GOING TO HAVE TO START   "
 7008 ? #6;"AGAIN.  PLEASE BE PATIENT AND    "
 7009 ? #6;"TURN ME OFF FOR AN INITIALIZING"
 7010 FOR I=1 TO 1000:NEXT I:STOP
 7020 POKE 87,0:POKE 88,P882:POKE 89,P892:CLOSE #1:POSITION 0,0
 7024 ? #6;"WELL...WE'RE IN TROUBLE............."
 7026 ? #6;"OUR OPEN DIDN'T WORK.  WILL YOU... "
 7027 ? #6;"LOOK AT OUR I/O DEVICE(S)?          "
 7028 ? #6;"IF IT'S A READ, DID WE HAVE DATA?"
 7029 ? #6;"I'M GOING TO START OVER........"
 7030 FOR I=1 TO 1000:NEXT I:GOTO 4
 7040 POKE 87,0:POKE 88,P882:POKE 89,P892:CLOSE #1:POSITION 0,0
 7044 ? #6;"GUESS WHAT...WE'VE GOT PROBLEMS (I.E. BUGS)"
 7046 ? #6;"OUR READ OR WRITE FAILED...YEP...FAILED"
 7047 ? #6;"DO WE HAVE A DISK IN THE TAPE READER?"
 7048 ? #6;"I'M GOING TO GIVE IT ANOTHER TRY."
 7049 ? #6;"PLEASE CHECK THE EQUIPMENT ...."
 7050 FOR I=1 TO 1000:NEXT I:GOTO 4
 7060 POKE 87,0:POKE 88,P882:POKE 89,P892:POSITION 0,0
 7064 ? #6;"THE CLOSE DIDN'T CLOSE PROPERLY       "
 7066 ? #6;"THIS MAY OR MAY NOT MEAN MUCH TO YOU"
 7067 ? #6;"BUT IT MEANS A LOT TO ME.  YEH....."
 7068 ? #6;"CHECK OUR I/O GRAR THAT COULD DO IT"
 7069 ? #6;"NOTHING FOR ME TO DO BUT START OVER"
 7070 FOR I=1 TO 1000:NEXT I:GOTO 4
 7080 REM TRAP FOR MAIN ROUTINE
 7082 POKE 87,0:POKE 88,P882:POKE 89,P892:POSITION 0,0
 7084 ? #6;"WHATEVER YOU DID REALLY BEFUDDLED ME  "
 7086 ? #6;"MUCH AS I HATE TO ADMIT CONFUSION  "
 7087 ? #6;"WE'RE JUST GOING TO HAVE TO START   "
 7088 ? #6;"AGAIN.  PLEASE BE PATIENT AND    "
 7089 ? #6;"TURN ME OFF FOR AN INITIALIZING"
 7090 FOR I=1 TO 1000:NEXT I:STOP
 9998 GRAPHICS 2+16:SETCOLOR 4,5,5:POSITION 0,5:? #6;"OFFICIAL TIME OUT!"
 9999 FOR I=0 TO 300:NEXT I:RETURN
10000 REM TRAP ROUTING FOR I/O
10010 GRAPHICS 5:GOSUB 830:POKE 752,1
10020 GOSUB 10030:GOTO 4
10030 POKE 87,0:POKE 88,P882:POKE 89,P892:POSITION 0,0
10040 ? #6;"UH...HI THERE...MY I/O IS IN TROUBLE  "
10050 ? #6;"WHILE YOU CHECK IT OUT, I'LL...    "
10060 ? #6;"INITIALIZE AND GET READY FOR MORE..."
10070 ? #6;"WORK ON THESE BLASTED TUTOUIALS  "
10080 ? #6;"WELL GOOD LUCK, HAVE A NICE DAY"
10090 REM COLOR 1:POKE 87,5:POKE 88,P88:POKE 89,P89
10100 FOR I=0 TO 500:NEXT I:RETURN
```

— EX5.5 —

```
4 REM EX5.5
5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE. WRITTEN BY ROBIN SHERER.
7 TRAP 10
10 DIM X(8):MENU=1000:MOV=10000:COLR=11000:POKE 623,1
20 FOR I=1 TO 4:X(I)=20+I*24:NEXT I:REM INITIAL POS OF P/M 1 TO 5
30 FOR I=5 TO 8:X(I)=110+I*9:NEXT I
40 TRAP 210:POKE 559,0
50 PRINT "}"
60 PC1=137:PC2=198:PC3=248:PC4=54:BAKC=0
70 POKE 704,PC1:POKE 705,PC2:POKE 706,PC3:POKE 707,PC4:PC5=28:REM SET UP INITIAL COLORS OF PLAYERS
80 A=PEEK(106)-8:POKE 54279,A:PMBASE=256*A:REM SET PLAYER-MISSLE STARTING ADDRESS AT A POINT 8*256(2048)
90 REM BYTES BELOW THE TOP OF RAM TO ALLOW ROOM AND BE SURE IT'S ON A 2K BOUNDRY.
100 REM POKE 559,46:POKE 53277,3:REM ENABLE PM GRAPHICS WITH 2-LINE THICKNESS &
110 POKE 53277,3
120 POKE 752,1:REM MAKE CURSOR INVISABLE UNTIL MOVED LATER
130 FOR I=PMBASE+384 TO PMBASE+1024:POKE I,0:NEXT I
140 FOR I=PMBASE+384 TO PMBASE+448:POKE I,255:NEXT I
150 FOR I=PMBASE+512 TO PMBASE+576:POKE I,255:NEXT I
160 FOR I=PMBASE+768 TO PMBASE+832:POKE I,255:NEXT I
170 FOR I=PMBASE+640 TO PMBASE+704:POKE I,255:NEXT I
180 FOR I=PMBASE+896 TO PMBASE+960:POKE I,255:NEXT I
190 POKE 53256,3:POKE 53257,3:POKE 53258,3:POKE 53259,3:POKE 53260,255:REM INITIAL SIZE OF PLAYERS
200 FOR I=1 TO 8:POKE 53247+I,X(I):NEXT I
210 POKE 559,46
220 X=50:Y=50:POKE 710,BAKC:POKE 752,1:? "":GOSUB 720
230 FOR I=1 TO 5:POKE 53247+I,X(I):NEXT I:REM MOVE PLAYERS ON SCREEN
235 REM ****** MAIN LOOP ********
240 S=STICK(0)
243 IF PEEK(53279)=3 THEN POKE 764,12:RUN "D:EX5.6"
245 IF PEEK(53279)=5 THEN GOSUB 1000
250 IF STRIG(0)=0 THEN GOSUB 380
260 IF S=15 THEN 240
270 X2=X1-(S=9)-(S=10)-(S=11)+(S=5)+(S=6)+(S=7)
280 Y2=Y1-(S=6)-(S=10)-(S=14)+(S=5)+(S=9)+(S=13)
290 IF X2<1 THEN X2=1
300 IF X2>37 THEN X2=37
310 IF Y2<1 THEN Y2=1
320 IF Y2>10 THEN Y2=10
340 POSITION X1,Y1:? "  "
350 POSITION X2,Y2:? "*"
360 X1=X2:Y1=Y2
370 GOTO 240:REM LOOP FOR ANOTHER MOVE OR EXIT
```

CONTINUED

```
380 S=STICK(0)
400 IF S=15 THEN 380
410 IF X2<7 THEN 460
420 IF X2<13 THEN 510
430 IF X2<19 THEN 560
440 IF X2<25 THEN 610
445 IF X2<34 THEN 655
450 GOTO 660
460 PC1=PC1+(S=14)-(S=13)
470 IF PC1<2 THEN PC1=2:RETURN
480 IF PC1>255 THEN PC1=255:RETURN
490 POSITION 30,17:? "   ":POSITION 30,17:? PC1
500 POKE 704,PC1:RETURN
510 PC2=PC2+(S=14)-(S=13)
520 IF PC2<0 THEN PC2=0:RETURN
530 IF PC2>255 THEN PC2=255:RETURN
540 POSITION 30,18:? "   ":POSITION 30,18:? PC2
550 POKE 705,PC2:RETURN
560 PC3=PC3+(S=14)-(S=13)
570 IF PC3<0 THEN PC3=0:RETURN
580 IF PC3>255 THEN PC3=255:RETURN
590 POSITION 30,19:? "   ":POSITION 30,19:? PC3
600 POKE 706,PC3:RETURN
610 PC4=PC4+(S=14)-(S=13)
620 IF PC4<0 THEN PC4=0:RETURN
630 IF PC4>255 THEN PC4=255:RETURN
640 POSITION 30,20:? "   ":POSITION 30,20:? PC4
650 POKE 707,PC4:RETURN
655 PC5=PC5+(S=14)-(S=13)
656 IF PC5<0 THEN PC5=0:RETURN
657 IF PC5>255 THEN PC5=255:RETURN
658 POSITION 30,21:? "   ":POSITION 30,21:? PC5
659 POKE 711,PC5:RETURN
660 BAKC=BAKC+(S=14)-(S=13)
670 IF BAKC<0 THEN BAKC=0:RETURN
680 IF BAKC>255 THEN BAKC=255:RETURN
690 POSITION 30,22:? "   ":POSITION 30,22:? BAKC
700 POKE 710,BAKC:RETURN
710 GOTO 380
720 POSITION 10,15:? "PLAYER COLORS":POSITION 10,16:? "**************"
730 POSITION 10,17:? "PLAYER 0(POKE 704)= ";PC1
740 POSITION 10,18:? "PLAYER 1(POKE 705)= ";PC2
750 POSITION 10,19:? "PLAYER 2(POKE 706)= ";PC3
760 POSITION 10,20:? "PLAYER 3(POKE 707)= ";PC4
765 POSITION 10,21:? "PLAYER 4(POKE 711)= ";PC5
770 POSITION 10,22:? "BACKGRD (POKE 710) = ";BAKC
780 RETURN
1000 REM ***** ENABLE 5TH PLAYER *******
1010 IF FLAG=0 THEN POKE 623,17:FLAG=1:GOTO 1030
1020 IF FLAG=1 THEN POKE 623,1:FLAG=0
1030 FOR I=5 TO 8:X(I)=110+I*8:NEXT I
1040 FOR I=5 TO 8:POKE 53247+I,X(I):NEXT I
1050 RETURN
10000 GRAPHICS 10:GOTO 10000
```

```
 4 REM EX 5.6
 5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE. WRITTEN BY ROBIN SHERER
 10 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:GOTO 410
 20 REM ***** PLOT SHAPE ******
 30 COLOR 0
 40 D=D+(ST=6)+(ST=7)+(ST=5)-(ST=9)-(ST=10)-(ST=11)
 50 E=E+(ST=13)+(ST=9)+(ST=5)-(ST=6)-(ST=10)-(ST=14)
 60 FOR X=0 TO 9:FOR Y=0 TO 25:PLOT X+D,Y+E:NEXT Y:NEXT X
 70 RESTORE 150:COLOR 1
 80 FOR N=1 TO 100:READ X:READ Y
 90 IF X=0 AND Y=0 THEN RETURN
 100 PLOT X+D,Y+E:NEXT N
 110 E$(1,60)="hhhJH0
  1KHHPwhJH0
  }1KPwhJH0FM%M
PhJ0fM%M
P'"
120 REM POKE 53248,30
140 SOUND 2,0,8,2:FOR VOL=1 TO 15 STEP 0.1:SOUND 0,25,4,VOL:SOUND 1,13,4,VOL
150 DATA 5,0,3,1,4,1,5,1,6,1,2,2,3,2,4,2,5,2,6,2,7,2,1,3,2,3,7,3,8,3,2,4,3,4,4,4,5,4,6,4,7,4,3,5,4,5,5,5,6,5
160 DATA 4,6,5,6,2,7,3,7,4,7,5,7,6,7,7,7,1,8,3,8,6,8,8,8,1,9,8,9,2,10,4,10,5,10,7,10,2,11,4,11,5,11,7,11
170 DATA 2,12,7,12,3,13,6,13,3,14,6,14,4,15,5,15,4,16,5,16,4,17,5,17,4,18,5,18,3,19,4,19,5,19,6,19
180 DATA 2,20,3,20,4,20,5,20,6,20,7,20,1,21,2,21,3,21,4,21,5,21,6,21,7,21,8,21,2,22,3,22,4,22,5,22,6,22,7,22
190 DATA 3,23,4,23,5,23,6,23,4,24,5,24,0,0,0,0
200 REM *****PLAYER SETUP *****
210 DIM E$(60):GRAPHICS 6:D=20:E=50:POKE 559,0:GOSUB 20:? "      WHICH IS THE PLAYER?":POKE 710,148:POKE 712,148
220 ? "SELECT = SIZES  OPTION = NEXT EX."
230 E$(1,60)="hhhJH0
 1KHHPwhJH0
 1KPwhJH0FM%M
PhJ0fM%M
P'"
240 A=PEEK(106)-16:POKE 54279,A:POKE 204,A+4:POKE 203,0:PMBASE=A*256
250 POKE 53277,3:POKE 704,40:POKE 53248,150:POKE 205,120:POKE 53256,0
260 FOR I=PMBASE+1024 TO PMBASE+1280:POKE I,0:NEXT I
270 RESTORE 290
280 FOR I=PMBASE+1155 TO PMBASE+1209:READ B:POKE I,B:NEXT I
290 DATA 8,8,60,60,126,126,195,195,126,126,60,60,24,24,126,126,165,165
300 DATA 129,129,90,90,90,90,66,66,36,36,36,36,24,24,24,24,24,24,24,24
310 DATA 60,60,126,126,255,255,126,126,60,60,24,24,0,0,0,0,0,0
320 POKE 559,62
330 REM
340 REM ******MAIN LOOP ******
350 REM
360 IF STRIG(0)=0 THEN GOSUB 20:FLAG=0
370 ST=STICK(0):S=PEEK(53279)
380 IF S=5 THEN GOSUB 670
390 IF S=3 THEN POKE 764,12:POKE 53256,0:RUN "D:EX5.7"
400 A=USR(ADR(E$),STICK(0)):GOTO 360
410 GRAPHICS 17:POKE 712,148
420 ? #6;" THIS NEXT EXAMPLE  "
430 ? #6;"  SHOWS YOU THE     "
440 ? #6;"DIFFERENCE BETWEEN A"
450 ? #6;"NORMALLY DRAWN SHAPE"
460 ? #6;"AND A PLAYER....use "
470 ? #6;"YOUR joystick TO SEE"
480 ? #6;"WHICH IS THE player!"
490 FOR I=1 TO 3000:NEXT I:GOTO 210
```


CONTINUED

```
500 REM ***** PLOT SHAPE ******
510 COLOR 1
520 D=D+(ST=6)+(ST=7)+(ST=5)-(ST=9)-(ST=10)-(ST=11)
530 E=E+(ST=13)+(ST=9)+(ST=5)-(ST=6)-(ST=10)-(ST=14)
540 RESTORE 370
550 FOR N=1 TO 100:READ X:READ Y
560 IF X=0 AND Y=0 THEN 110
570 PLOT X+D,Y+E:NEXT N
580 IF FLAG=0 THEN COLOR 0:GOTO 130
590 RETURN
600 FOR X=1 TO 8:FOR Y=0 TO 25:PLOT X+D,Y+E:NEXT Y:NEXT X
610 RETURN
620 DATA 5,0,3,1,4,1,5,1,6,1,2,2,3,2,4,2,5,2,6,2,7,2,1,3,2,3,7,3,8,3,2,4,3,4,4,4,5,4,6,4,7,4,3,5,4,5,5,5,6,5
630 DATA 4,6,5,6,2,7,3,7,4,7,5,7,6,7,7,7,1,8,3,8,6,8,8,8,1,9,8,9,2,10,4,10,5,10,7,10,2,11,4,11,5,11,7,11
640 DATA 2,12,7,12,3,13,6,13,3,14,6,14,4,15,5,15,4,16,5,16,4,17,5,17,4,18,5,18,3,19,4,19,5,19,6,19
650 DATA 2,20,3,20,4,20,5,20,6,20,7,20,1,21,2,21,3,21,4,21,5,21,6,21,7,21,8,21,2,22,3,22,4,22,5,22,6,22,7,22
660 DATA 3,23,4,23,5,23,6,23,4,24,5,24,0,0,0,0
670 REM ***** SIZES *****
680 ? :? :? "PRESS 0, 1, OR 3 FOR SIZE":? "SELECT = SIZES   OPTION = NEXT EX."
690 POKE 764,255
700 PK=PEEK(764)
710 IF PK=50 THEN POKE 53256,0:RETURN
720 IF PK=31 THEN POKE 53256,1:RETURN
730 IF PK=26 THEN POKE 53256,3:RETURN
740 GOTO 700
```

— EX5.7 —

```
4 REM EX 5.7
5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE(ROBIN SHERER)
10 GOTO 140
20 REM
30 REM ****** MAIN LOOP **********
40 REM
50 ST=STICK(0)
60 X=X+(ST=6)+(ST=7)+(ST=5)-(ST=9)-(ST=10)-(ST=11)
70 Y=Y+(ST=13)+(ST=9)+(ST=5)-(ST=6)-(ST=10)-(ST=14)
80 GOSUB 330
85 IF PEEK(53279)=3 THEN POKE 764,12:RUN "D:EX5.8"
90 IF PADDLE(0)=228 THEN 110
100 X=PADDLE(0)
110 POKE 53248,X
120 GOTO 50
130 REM
140 REM PMB=PLAYER MISSILE BASE ADDRESS IN PAGES(256 BYTES PER PAGE)
150 REM
160 POKE 704,40
170 PMB=PEEK(106)-16
180 POKE 54279,PMB
190 PMBASE=PMB*256
200 POKE 53277,3
```

CONTINUED

```
210 X=125:Y=100:YSAVE=100
220 POKE 53256,0
230 FOR I=PMBASE+1024 TO PMBASE+1280:POKE I,0:NEXT I
240 RESTORE 210:CNT=0
250 FOR I=PMBASE+1024+Y TO PMBASE+1280
260 READ B:IF B=0 THEN 310
270 CNT=CNT+1
280 POKE I,B
290 NEXT I
300 DATA 8,60,126,195,126,60,24,126,165,0,0
310 POKE 559,62
320 GOTO 50
330 REM
340 REM ***** UP/DOWN MOVE *******
350 IF YSAVE=Y THEN RETURN
360 IF YSAVE<Y THEN 430
370 RESTORE 210
380 FOR I=1 TO CNT
390 READ B
400 POKE PMBASE+1024+Y+I,B
410 NEXT I
420 YSAVE=Y:POKE PMBASE+1024+Y+CNT+1,0:RETURN
430 RESTORE 210
440 FOR I=1 TO CNT
450 READ B
460 POKE PMBASE+1024+Y+I-1,B
470 NEXT I
480 YSAVE=Y:POKE PMBASE+1024+Y-1,0:RETURN
```

— EX5.8 —

```
4 REM EX5.8
5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE(ROBIN SHERER)
 SOFTWARE(ROBIN SHERER)
10 GOTO 30
20 FOR I=PMBASE+1024 TO PMBASE+1536:POKE I,0:NEXT I:RETURN
30 GOSUB 200:REM DRAW MOUNTAINS
40 DIM E$(60)
50 GOSUB 300:REM DRAW STARS
60 POKE 710,0:POKE 712,0:POKE 704,123
70 GOSUB 380:REM SETUP PLAYERS
80 POKE 204,A+4:POKE 203,0:POKE 205,120
90 E$(1,60)="hhhJH0
 1KHHPwhJH0
 }1KPwhJH0FM%M
PhJ0fM%M
p'"
100 REM POKE 53248,30
110 REM ***MAIN LOOP *********
120 SOUND 2,0,8,2:FOR VOL=1 TO 15 STEP 0.1:SOUND 0,25,4,VOL:SOUND 1,13,4,VOL
130 A=USR(ADR(E$),STICK(0))
140 NEXT VOL
150 FOR VOL=14 TO 0 STEP -0.1:SOUND 0,25,4,VOL:SOUND 1,13,4,VOL
160 A=USR(ADR(E$),STICK(0))
170 NEXT VOL
180 IF PEEK(53279)=3 THEN POKE 764,12:RUN "D:EX5.9"
190 GOTO 110
```


CONTINUED

```
200 REM **** PLOT BACKGROUND *****
210 GRAPHICS 8+16:RESTORE 260
220 COLOR 1
230 READ X,Y:PLOT X,Y
240 READ X,Y:IF X>900 THEN RETURN
250 DRAWTO X,Y+10:GOTO 240
260 DATA 0,157,27,166,39,166,49,166,57,157,57,158,69,158,74,164,87,170,87,171,90,169,95,140
270 DATA 110,70,112,65,114,50,116,63,127,93,130,97,140,97,153,63,159,57,165,84,168,91,175,103,181,123
280 DATA 190,174,199,174,205,165,207,149,229,133,245,156,256,133,271,120,283,165,287,172,291,164,300,89,305,76
290 DATA 310,63,999,999
300 REM **** STARS *****
310 TRAP 370
320 RESTORE 340
330 READ X,Y:PLOT X,Y:GOTO 330
340 DATA 2,4,5,8,9,10,13,43,34,65,7,3,23,66,123,45,300,67,301,65,246,34,234,49,261,68,241,133,123,55,67,98
350 DATA 12,64,75,58,119,10,133,43,4,65,7,93,203,66,12,45,178,67,145,65,46,34,187,49,123,68,21,133,243,55,116,74
360 DATA 123,5,156,7,213,8,295,12,296,34,294,56,230,56,230,6,78,9
370 RETURN
380 REM **** SETUP PLAYERS ****
390 A=PEEK(106)-40:POKE 54279,A:PMBASE=A*256
400 GOSUB 20:REM CLEAR PM AREA
410 POKE 53277,3:REM TURN ON PM DMA
420 Z=20:REM INITIAL Y POSITION OF SHIP
430 X=150:REM INITIAL X POSITION OF SHIP
440 POKE 53248,X:POKE 53249,X:REM POSITION SHIP (PLAYER0) AND FLAME (PLAYER1)
450 RESTORE 480
460 FOR I=PMBASE+1024+Z TO PMBASE+1280:READ SHAPE:IF SHAPE<>0 THEN POKE I,SHAPE:NEXT I
470 POKE 623,1:REM SET PRIORITY
480 DATA 8,60,126,195,126,60,24,126,165,129,90,0,0,0
490 POKE 559,62
500 RETURN
```

— EX5.10 —

```
3 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE(ROBIN SHERER)
5 REM EX5.10
10 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:POKE 559,0:GOTO 30
20 FOR I=PMBASE+512 TO PMBASE+1280:POKE I,0:NEXT I:RETURN
30 GRAPHICS 0:? :? :? " PRESS OPTION TO GO ON":POKE 752,1
40 ? :? " PRESS SELECT FOR SINGLE LINE          RESOLUTION PLAYER"
50 ? :? " PRESS START FOR DOUBLE LINE           RESOLUTION PLAYER"
70 POKE 704,40
80 PMB=PEEK(106)-16
90 POKE 54279,PMB
100 PMBASE=PMB*256
110 POKE 53277,3:GOSUB 20:POKE 559,34
120 X=125:Y=110:YSAVE=100
130 POKE 53256,0:GOTO 200
140 REM ****** MAIN LOOP *******
150 P=PEEK(53279)
160 IF P=5 THEN POKE 559,62:POKE 53248,100
170 IF P=6 THEN POKE 559,46:POKE 53248,100
180 IF P=3 THEN POKE 764,12:RUN "D:EX5.11"
190 GOTO 150
```

CONTINUED

```
200 REM
210 REM SINGLE LINE RESOLUTION ********
220 REM
230 RESTORE 290:CNT=0
240 FOR I=PMBASE+1024+Y TO PMBASE+1280
250 READ B:IF B=0 THEN 300
260 CNT=CNT+1
270 POKE I,B
280 NEXT I
290 DATA 124,214,214,254,124,68,68,204,0,0,0
300 REM
310 REM DOUBLE LINE RESOLUTION ********
320 REM
330 RESTORE 290:CNT=0
340 FOR I=PMBASE+512+Y TO PMBASE+640
350 READ B:IF B=0 THEN 390
360 CNT=CNT+1
370 POKE I,B
380 NEXT I
390 GOTO 150
```

— EX5.11 —

```
4 REM EX5.11
5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE(ROBIN SHERER)- PACMAN TM OF ATARI INC.
10 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:GOTO 240
20 FOR I=PMBASE+1024 TO PMBASE+1792:POKE I,0:NEXT I:RETURN
30 REM ******PLAYER SETUP ******
40 GRAPHICS 7:D=20:E=50:POKE 710,0:POKE 712,0:POKE 704,41:POKE 706,41:POKE 559,62
45 COLOR 2:PLOT 1,57:DRAWTO 159,57:PLOT 1,65:DRAWTO 159,65
50 A=PEEK(106)-32:POKE 54279,A:POKE 204,A+4:POKE 203,0:PMBASE=A*256
60 POKE 53277,3:X=150:POKE 205,120:POKE 53256,0:POKE 53250,150:POKE 53258,0
70 GOSUB 20
80 RESTORE 110:Y=150
90 FOR I=PMBASE+1024+Y TO PMBASE+1209:READ B:IF B<>0 THEN POKE I,B:NEXT I
100 REM ****** DATA FOR PACMAN1 ******
110 DATA 28,62,119,127,254,252,248,252,254,127,126,62,28,0,0
120 RESTORE 150
130 FOR I=PMBASE+1536+Y TO PMBASE+1792:READ B:IF B<>0 THEN POKE I,B:NEXT I
140 REM ****** DATA FOR PACMAN2 ******
150 DATA 28,62,118,255,255,255,255,255,126,126,62,28,0,0,0,0
160 POKE 559,62
210 ST=STICK(0):S=PEEK(53279)
220 IF S=3 THEN POKE 764,12:RUN "D:EX5.12"
222 IF ST=15 THEN 210
225 X=X+(ST=7)-(ST=11):POKE 53248,X:POKE 53250,0:FOR J=1 TO 50:NEXT J
226 POKE 53250,X:POKE 53248,0
230 GOTO 210
240 GRAPHICS 17:POKE 712,69:POKE 710,19
250 ? #6;"ANIMATION OF SHAPES "
260 ? #6;"  IS AS SIMPLE AS   "
270 ? #6;"SWITCHING BETWEEN   "
280 ? #6;" TWO PLAYERS AS YOU "
290 ? #6;"MOVE AROUND.....use "
300 ? #6;"YOUR JOYSTICK TO SEE"
310 ? #6;"THE pacman(TM) MOVE!"
320 FOR I=1 TO 3000:NEXT I:GOTO 40
330 POKE 764,255
```

```
4 REM EX5.12
5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE(ROBIN SHERER)- PACMAN TM OF ATARI INC.
10 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:GOTO 270
20 FOR I=PMBASE+1024 TO PMBASE+1536:POKE I,0:NEXT I:RETURN
30 REM *****PLAYER SETUP *****
40 GRAPHICS 7:POKE 710,0:POKE 712,69:POKE 704,41:POKE 706,41:POKE 559,62
45 POKE 708,103:POKE 752,1:POKE 709,139:POKE 710,100
47 COLOR 1:FOR X=20 TO 150 STEP 30:PLOT X,61:PLOT X+1,61:NEXT X
50 COLOR 2:PLOT 1,57:DRAWTO 159,57:DRAWTO 159,66:DRAWTO 1,66:DRAWTO 1,57
52 PLOT 157,57:DRAWTO 157,66:PLOT 158,57:DRAWTO 158,66
55 COLOR 3:FOR J=1 TO 4:FOR K=1 TO 5
56 T=K+J*30
57 PLOT T,57:DRAWTO T,66:NEXT K:NEXT J
60 A=PEEK(106)-32:POKE 54279,A:POKE 204,A+4:POKE 203,0:PMBASE=A*256
70 POKE 53277,3:X=50:POKE 205,120:POKE 53256,0:POKE 53250,150
80 GOSUB 20
90 RESTORE 120:Y=150:ERAX=-10
100 FOR I=PMBASE+1024+Y TO PMBASE+1280:READ B:IF B=999 THEN 110
105 POKE I,B:NEXT I
110 REM ***** DATA FOR PACMAN1 *****
120 DATA 28,62,119,127,254,252,248,252,254,127,126,62,28,999
130 RESTORE 160
140 FOR I=PMBASE+1536+Y TO PMBASE+1792:READ B:IF B=999 THEN 150
145 POKE I,B:NEXT I
150 REM ***** DATA FOR PACMAN2 ******
160 DATA 28,62,118,255,255,255,255,255,126,126,62,28,999
170 POKE 559,62
180 REM
190 REM ******MAIN LOOP ******
200 REM
205 POKE 53278,1:REM CLEAR ALL COLISION LOCATIONS (SO THEY HOLD 0'S)
220 ST=STICK(0):S=PEEK(53279)
230 IF S=3 THEN POKE 764,12:RUN "D:EX5.13"
240 X=X+(ST=7)-(ST=11):POKE 53248,X:POKE 53250,0:FOR J=1 TO 50:NEXT J
242 IF PEEK(53252)<>0 THEN GOSUB 1000:GOTO 220
245 IF X>220 THEN X=40
250 POKE 53250,X:POKE 53248,0
260 GOTO 220
270 GRAPHICS 17:POKE 712,245
280 ? #6;"NOW WE ADD CHECKING "
290 ? #6;"FOR COLLISIONS WITH "
300 ? #6;" THE WALLS AND ALSO "
310 ? #6;" PRIORITY WITH THE  "
320 ? #6;"OTHER SHAPES....use "
330 ? #6;"YOUR JOYSTICK TO SEE"
340 ? #6;"THE pacman(TM) MOVE "
350 ? #6;"  IN AND OUT OF THE "
360 ? #6;"   TUNNEL SHAPES   "
365 FOR I=1 TO 2000:NEXT I
370 GOTO 30
1000 REM **** COLLISION SOUND ****
1002 COL=PEEK(53252)
1004 IF COL=2 THEN 1015
1005 IF COL<>1 THEN 1030
1006 ? "YUMMY!!";? :?
1007 FOR I=200 TO 0 STEP -1:SOUND 0,I,10,8:NEXT I:? :? :?
1008 COLOR 0:ERAX=ERAX+30:PLOT ERAX,61:PLOT ERAX+1,61
1009 X=X+8:POKE 53248,X:POKE 53250,X
1010 GOTO 1030
1015 SOUND 0,50,8,8
1020 FOR I=1 TO 30:NEXT I:SOUND 0,0,0,0
1030 POKE 53278,1:RETURN
```

```
4 REM EX5.13
5 REM (C) 1982 BY SANTA CRUZ EDUCATIONAL SOFTWARE(ROBIN SHERER)
10 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:GOTO 240
20 FOR I=PMBASE+1024 TO PMBASE+1792:POKE I,0:NEXT I:RETURN
30 REM *****PLAYER SETUP *****
40 GRAPHICS 7:POKE 710,0:POKE 712,0:POKE 704,41:POKE 706,41:POKE 559,62
50 A=PEEK(106)-24:POKE 54279,A:POKE 204,A+4:POKE 203,0:PMBASE=A*256
60 POKE 53277,3:X=150:POKE 205,120:POKE 53256,0:POKE 53250,150
70 GOSUB 20
80 RESTORE 110:Y=150
90 FOR I=PMBASE+1024+Y TO PMBASE+1209:READ B:IF B<>0 THEN POKE I,B:NEXT I
100 REM ***** DATA FOR FIRST HALF ***
110 DATA 2,2,2,2,2,2,2,2,4,4,4,8,112,0,0,0
120 RESTORE 150
130 FOR I=PMBASE+1536+Y TO PMBASE+1792:READ B:IF B<>0 THEN POKE I,B:NEXT I
140 REM ***** DATA FOR 2ND HALF *****
150 DATA 160,160,160,160,160,160,160,160,160,144,144,144,136,135,0,0,0
160 POKE 559,62
165 ? "         16 BIT WIDE SHAPE"
167 POKE 752,1
168 ? "     <==== USE JOYSTICK =====>"
170 REM
180 REM ******MAIN LOOP ******
190 REM
200 IF STRIG(0)=0 THEN GOSUB 20:FLAG=0
210 ST=STICK(0):S=PEEK(53279)
220 IF S=3 THEN POKE 764,12:RUN "D:EX5.14"
225 X=X+(ST=7)-(ST=11):POKE 53248,X:POKE 53250,X+8
230 GOTO 200
240 GRAPHICS 17
250 ? #6;"  DO YOU WANT MORE  "
260 ? #6;"COMPLICATED SHAPES? "
270 ? #6;"JUST USE TWO PLAYERS"
280 ? #6;"NEXT TO EACH OTHER.."
290 ? #6;"...this example is  "
300 ? #6;"almost the same as  "
310 ? #6;"before, but....."
320 FOR I=1 TO 3000:NEXT I:GOTO 40
```